Tentamen på kurs: **DA339A, Objektorienterad programmering**

Provkod: 2002 Tentamen 2, 2,5 hp

220821 kl 8.15 – 13.15

Tillåtna hjälpmedel:

- **Inga tillåtna hjälpmedel utöver det som finns i den trycka tentamen, utdelat blad med uppgiftspoäng och eventuell medtagen engelsk-svensk ordbok eller annan språkkombination (anteckningar får inte förekomma i medtagna ordböcker)**
- **Anteckningar och svar får endast göras på papper tillhandahållna av tentamensorganisationen (undantag gäller för studenter som svarar digitalt i dokument på dator men inga egna medtagna papper får användas under tentan).**

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 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.
- **Fråga 1 besvaras på specifik svarsblankett.** Övriga frågor besvaras på blanka, linjerade eller rutade papper som tillhandahålls av tentamensvakter.

Lärare besöker tentamen för frågor i samband med start samt vid ungefär 10.00 och 11.30.

Instruktioner för detta extra tillfälle

Tentamen är en extra tenta för att komplettera den Tentamen 2 230811 som avbröts på grund av brandlarm.

Du ska ha fått ett papper av lärare vid start som visar hur många poäng du uppnått på respektive fråga på tillfället 230811. Frågor med samma nummer på tentorna 230811 och 230821 anses motsvara varandra och har samma maxpoäng.

Frågor som du fått full poäng på ska du inte göra om 230821.

Om du lämnar in en uppgift på tillfället 230821 ersätter denna motsvarande uppgift från 230811, oavsett om du får högre eller lägre poäng 230821.

Det är du som avgör vilka uppgifter du vill lämna in och ersätta eller komplettera med om du inte lämnade in något alls för uppgiften 230811. Detta innebär att den nuvarande poängen för uppgiften stryks och ersätts med den poäng du erhåller vid rättning av den nya inlämnade uppgiften. Uppgifter som inte lämnades in 230811 räknas som att de har fått 0 poäng i nuläget.

Exempel: Du har på tillfället 230811 fått 2 poäng av 4 möjliga på uppgift 7. Du lämnar in en lösning för uppgift 7 230821 och denna bedöms vid rättning få 3 poäng. De 2 poäng du fick 230811 byts då ut mot 3 poäng för uppgift 7. Din totala poäng på tentan ökar med 1 poäng.

Exempel: Du har på tillfället 230811 fått 2 poäng av 4 möjliga på uppgift 7. Du lämnar in en lösning för uppgift 7 230821 och denna bedöms vid rättning få 1 poäng. De 2 poäng du fick 230811 byts då ut mot 1 poäng för uppgift 7. Din totala poäng på tentan minskar med 1 poäng.

Exempel: Du har på tillfället 230811 inte lämnat in uppgift 7 (har nu 0 poäng). Du lämnar in en lösning för uppgift 7 230821 och denna bedöms vid rättning få 1 poäng. Din totala poäng på tentan ökar med 1 poäng.

Du kan inte göra om alla uppgifter och endast räkna den version av uppgiften med bäst poäng.

Du måste avgöra vilka uppgifter du vill göra om.

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 i svarsblankett.

Fråga	Påstående

A	Antag att metoden nedan finns i en klass:
	`public void methodA (String[] stringArray){`
	`  /*kod*/`
	`}`
	Kan metoden anropas från en annan metod så här:
	`String s = methodA(sArray[]);`
	Svar: FALSK

B	Nyckelordet **final** betyder att en variabel, metod eller klass kan modifieras. Svar: SANT
C	Om en klass är abstrakt kan man inte skapa objekt av klassen. Svar: SANT
D	En abstrakt metod måste implementeras av alla subklasser till superklassen som innehåller den abstrakta metoden. Svar: SANT
E	Om man inte definierar metoden `toString()` finns metoden ändå tillgänglig. Detta beror på att den ärvs av klassen `Object` som är superklass till alla klasser. Svar: SANT

F	Antag att klassen B är en subklass till klassen A. Klassen B har en metod methodB.
	Kan man skriva:
	`A a = new B();`
	`a.methodB();`
	Svar: FALSK

G	Ett interface kan ha publika och privata instansvariabler samt konstanter. Svar: FALSK
H	Vid hantering av undantag, kan det finnas endast ett **catch** och ett **finally** block men flera **try** block. Svar: FALSK, endast ett try, men kan ha många catch
I	En klass kan ärva från en annan klass och kan implementera flera interfacer. Svar: SANT
J	Antag att klassen B är en subklass till klassen A. B klassens egna konstruktor kommer alltid anropas och exekveras först och sen exekveras A klassens konstruktor. Svar: FALSK
K	Ett interface innehåller vanligtvis implementation av metoder. Svar: FALSK
L	En enum är en klasstyp och kan instansieras, dvs. det går att skapa ett objekt av en enum med nyckelordet new. Svar: FALSK

M	Ett interface kan inte implementera ett annat interface men kan dock ärva ett annat interface. Svar: SANT
N	Polymorfism innebär att ett objekt kan bete sig som om det tillhör två olika klasser. Svar: SANT
O	En bra tumregel för generalisering är om förhållandet mellan klasserna A och B går att uttrycka som ”A har en B”. Svar: FALSKT
P	Vid generalisering ärvs även associationer som superklassen har till andra klasser. Svar: SANT
Q	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. Svar: SANT
R	Navigeringsriktning på en association visar vilken klass som anropar den andra klassen i associationen. Svar: SANT
S	En use-association är en starkare association mellan två klasser än en vanlig enkel association. Svar: FALSKT
T	En klass kan ha en association till sig själv Svar: SANT

Uppgift 2 2p

Förklara kortfattat begreppet BCE Boundary-Control-Entity och varför detta är relevant att använda vid mjukvaruutveckling. Frågan bör kunna besvaras med 6-8 meningar.

Svar: Se föreläsningar och slides

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.

Svar: Se föreläsningar och slides

Uppgift 4 2p

Vad blir utskriften när metoden sumLetters exekveras?

```
public String methodA(int a, int b){

    String text = ("#111: text = " + (a + b) + "c");

    return text;

}

public String methodA(double a, double b){

    String text = ("#222: text = " + (a - b) + "c");

    return text;

}

public void sumMethodA(){

    double a = 2;

    double b = 2;

    System.out.println(methodA(a, b));

}
```

Exempel på hur metoden kan anropas och köras:

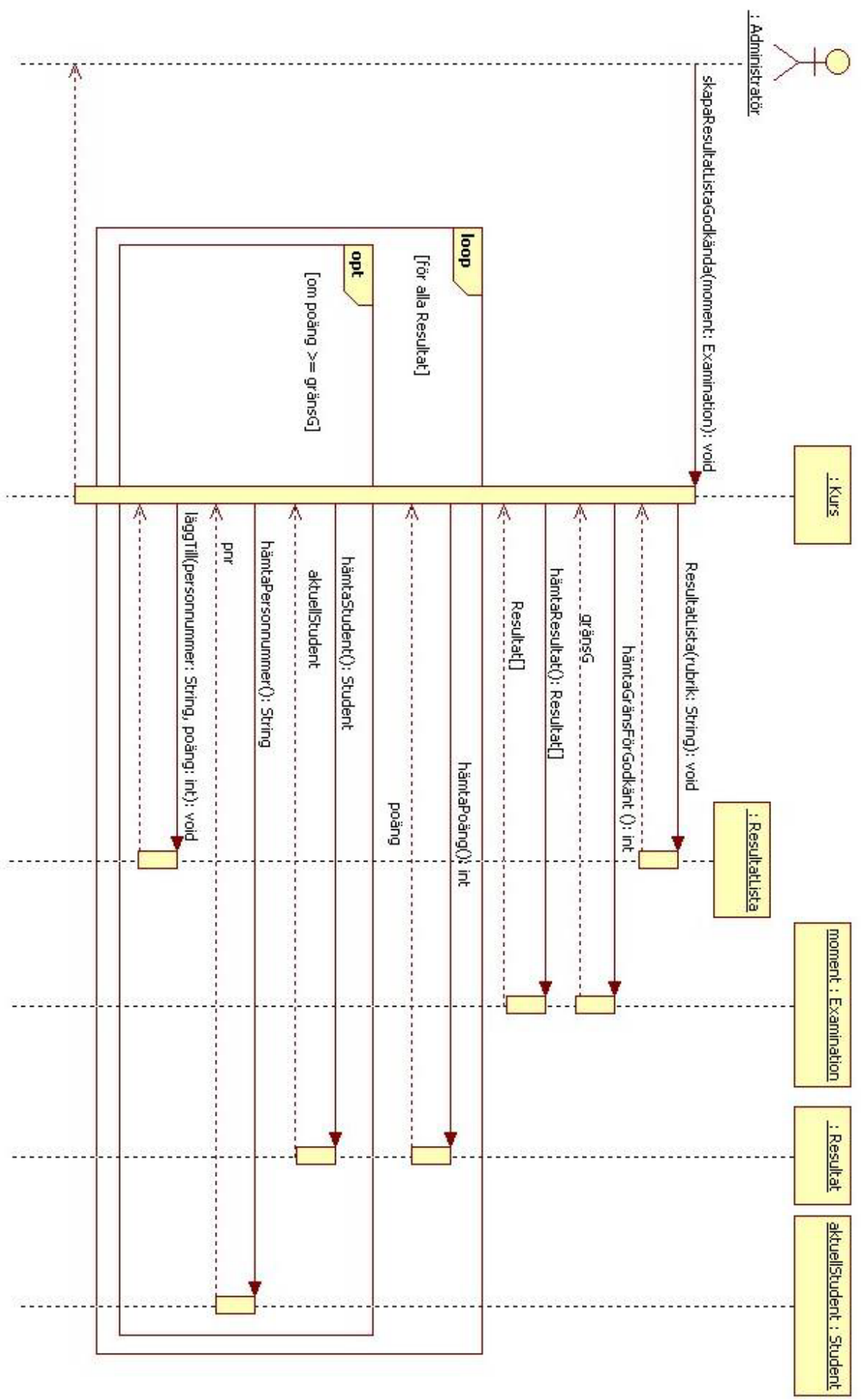
```
public class Main {
   Overloading test = new Overloading();
   test.sumMethodA();
}
```

Svar/Lösningsförslag
#222: text = 0.0c

Uppgift 5 4p

På nästa finns en bild med ett sekvensdiagram. I sekvensdiagrammet finns information om klasser i from av klassnamn och metoder i dessa klasser samt viss information om vad som sker i metoderna vid körnings av programmet. Skriv den kod som går att utläsa ur diagrammet för klasserna Kurs, Examination och Resultat.

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


: Administratör
skapaResultatListaGodkända(moment: Examination): void
: Kurs
loop
[För alla Resultat]
opt
[om poäng >= gränsG]
ResultatLista(rubrik: String): void
: ResultatLista
hämtaGränsFörGodkänt (): int
gränsG
hämtaResultat(): Resultat[]
Resultat[]
moment : Examination
hämtaPoäng(): int
poäng
: Resultat
hämtaStudent(): Student
aktuellStudent
aktuellStudent : Student
hämtaPersonnummer(): String
pnr
läggTill(personnummer: String, poäng: int): void

Lösningsförslag

```
class Kurs{
  void skapaResultatListaGodkända(Examination moment){

    lista = new resultatLista(namn);
    gränsG = momnet.hämtagränsFörGodkänt();
    resultat = moment.hämtaResultat()

    for(alla resultat){
      poäng = resultat[x].hämtaPoäng();

      if(poäng>= gränsG){
        aktuellStudent = resultat[x].hämtaStudent();
        pnr = aktuellStudent.hämtaPersonnummer();
        lista.läggTill(pnr, poäng);
      }
    }
  }
}

class Examination{
  int hämtaGränsFörGodkänt(){…}
  Resultat[] hämtaResultat(){…}
}

class Resultat{
  int hämtaPoäng(){…}
  Student hämtaStudent(){…}
}
```

Uppgift 6 2p

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

```
String stringNumber = "AAA111";

public double convertToNumber()
 {
    double number = Double.parseDouble(stringNumber);
    return number;
 }
```

Svar/Lösningsförslag:

```
String stringNumber = "AAA111";
String answerToException = "Could convert: ";

public double convertToNumber()
 {
          double number = 0;
          try
          {
              number = Double.parseDouble(stringNumber);
          }
          catch (NumberFormatException e)
          {
              answerToException = "Could not convert: ";
          }
          finally
          {
              System.out.println(answerToException + stringNumber + " to a
number");
          }
          return number;
 }
```

Uppgift 7 4p

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

```
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;
     }
}

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;
    }
}
```

Uppgift 7a 1p

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

Uppgift 7b 2p

Skapa en enda konstruktor för klassen BeachParty med nödvändiga parametrar för att initiera alla instansvariabler i klassen BeachParty och dess superklass.

Uppgift 7c 1p

Skriv en toString-metod i klassen BeachParty så att resultatet nedan erhålls. En testkörning av metoden i Main ger resultatet:

```
public class Main {
     public static void main(String[] args) {
       BeachParty bParty = new BeachParty("Best Party", 20230821,
"Malmö", "Kranen street", "Tenta", "Basse");
        System.out.println(bParty);
     }
 }
```

Output:

```
Name: Best Party | Created: 20230821 | Adress: Malmö, Kranen street |
Theme: Tenta | DJ playing: Basse
```

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

Lösningsförslag:

```
public class BeachParty extends Event {
    private String theme;
    private String playingDJ;
  public BeachParty(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

Konstruera ett lämpligt klassdiagram utifrån systembeskrivningen nedan. Identifiera lämpliga model-klasser för systemet utifrån beskrivningen (model från begreppet MVC). 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 – du förväntas använda minst en generalisering och minst en komposition eller aggregation i din lösning.

Skriv ut namn på alla associationer för att förtydliga dessa. Om det går att ange multiplicitet för en association ska detta finnas – gör eventuella lämpliga antaganden där detta kan vara rimligt. Gör inga tillägg med information om delar av systemet som inte finns i systembeskrivningen.

Det är inte nödvändigt att visa attribut eller operationer i klassdiagrammet. Det räcker att skriva ut klassnamn, associationer med namn och multiplicitet. Korrekt UML-notation ska användas (se Bilaga 1).

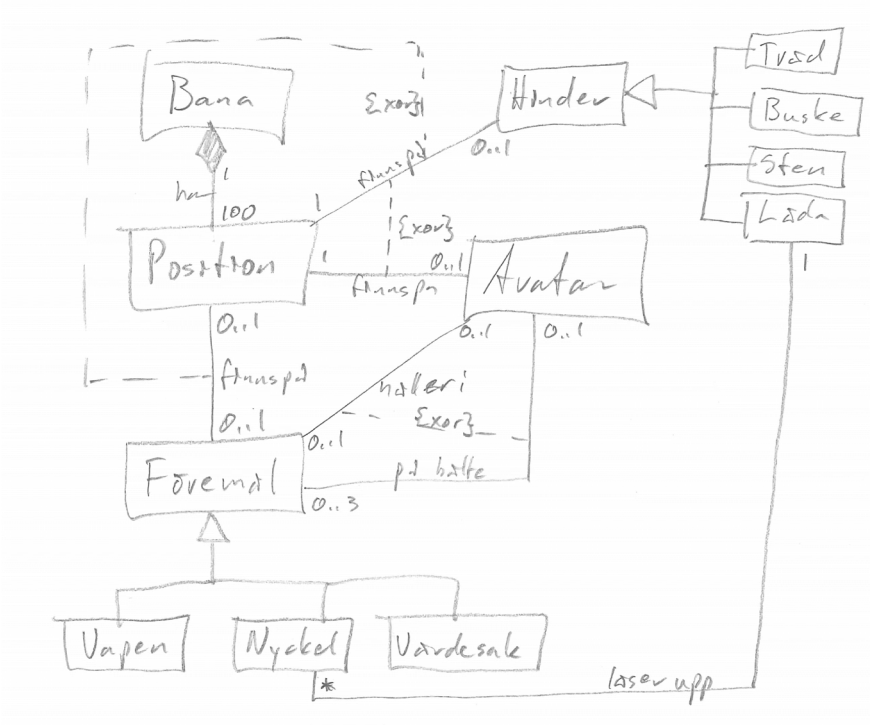
Systembeskrivning

Ett spel består av ett antal banor som utgörs av olika banor som spelaren måste ta sin avatar genom. Varje bana utgörs av en spelplan med 100 positioner. Avatarer i spelet kan befinna sig på en position i taget.

På en viss position kan det också finnas föremål som spelaren kan låta avataren plocka upp. En avatar kan bara hålla i ett föremål i taget. Avataren kan dock hänga upp till 3 föremål på sitt bälte. Föremål kan vara saker spelaren kan använda sig av, exempelvis vapen eller nycklar som låser upp lådor. Föremål kan också vara värdesaker man samlar ihop för mer poäng. En avatar och ett föremål kan ha samma position. Det kan dock inte finnas två avatarer eller två föremål på samma position.

En position kan vara upptagen på grund av att den innehåller någon typ av objekt som inte kan tas upp och då kan ingen avatar eller annat föremål finnas på denna position. Objekt som kan uppta en position är lådor, stenar, träd eller buskar. Vissa saker går att förstöra på olika sätt – ett träd kan huggas ner med en yxa (en sorts vapen) och en buske kan rivas upp. Stenar och lådor går inte att förstöra. Lådor kan öppnas med en nyckel som passar till just den lådan.

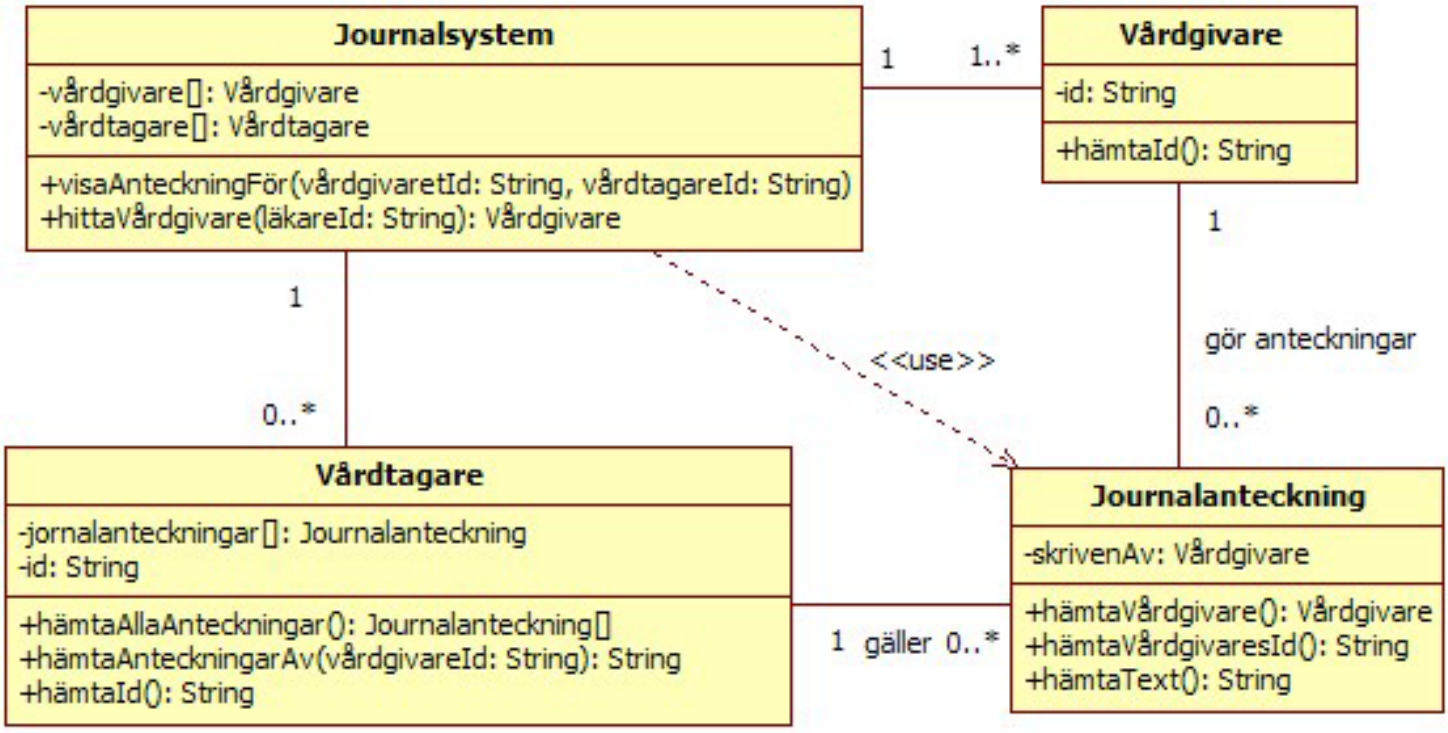
Lösningsförslag:



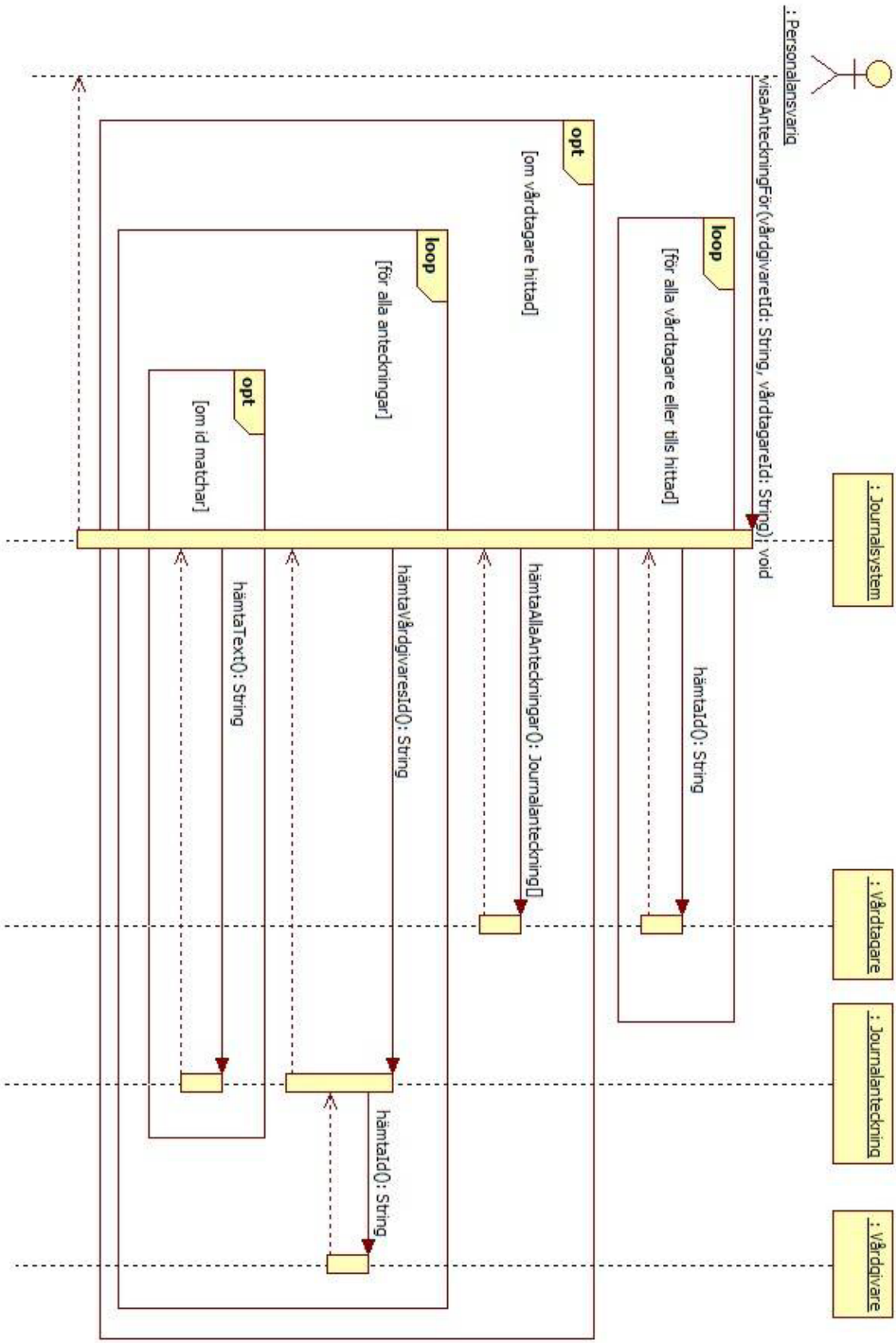
Uppgift 9 7p

Klassdiagrammet nedan beskriver ett system som hanterar journalanteckningar som görs på ett vårdboende. Givet klassdiagrammet nedan rita ett sekvensdiagram som visar vad som sker när personalansvarig vill ta fram alla anteckningar som en given vårdgivare gjort gällande en given vårdtagare.

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 personalansvarig som anropar metoden visaAnteckningarFör(….).



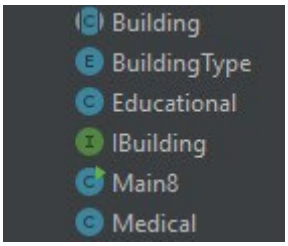
Lösningsförslag:


: Personalansvarig
visaAnteckningFör(vårdgivarenId: String, vårdtagarenId: String): void
opt
[om vårdtagare hittad]
loop
[för alla vårdtagare eller tills hittad]
loop
[för alla anteckningar]
opt
[om id matchar]
: Journalsystem
: Vårdtagare
: Journalanteckning
: Vårdgivare
hämtaId(): String
hämtaAllaAnteckningar(): Journalanteckning[]
hämtaVårdgivaresId(): String
hämtaText(): String
hämtaId(): String

Uppgifter för betyg VG

Uppgift 10a-d

Eumerationen **BuildingType** är given nedan. Du ska skriva en klass **Building** som använder enumerationen och implementerar interfacet **IBuilding**. Senare i uppgiften skall du också skriva ett par konkreta subklasser till Building. 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 BuildingType {
   Medical,
   Residential,
   Educational
}
```

Uppgift 10a: Interface

Komplettera interfacet vid markeringarna 1 till 3.

```
public interface IBuilding {
   //En byggnad (Building) måste ha en typ av enum-typen
BuildingType.
   //1. Komplettera med en getter- och en setter-metod som
   //   subklasserna (i detta fall klassen Building) ska
    //   implementera.
   //En byggnad (Building) skall räkna ut och returnera sin
    //area utifrån sin längd och bredd.
        //- Area = längd*bredd (l*b)
   //2. Komplettera med metod som returnerar arean som en double.
    //3. Skriv en metod getBuildingInfo som skall returnera en
String.
    //   (mer om denna metod i uppgift 10b).
}
```

Uppgift 10b: Abstrakt klass och abstrakt metod

Skriv en abstrakt klass **Building** som implementerar interfacet ovan. Klassen skall innehålla all kod som implementation av interfacet kräver, med undantag av metoden **getBuildingInfo** som skall definieras som en abstrakt metod i klassen. Du får deklarera variabler och skriva flera metoder 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 **getBuildingInfo** i klassen Building som skall returnera en string.
I denna del, skriv bara de ändringar som du behöver göra i klassen Building.

Uppgift 10c: Konkreta klasser

Skriv två klasser, **Educational** och **Medical** som subklasser till Building. Deklarera en valfri instansvariabel i varje klass. Ifall du inte kommer på något bra attribut, använd följande:
Klassen Educational: numOfClassRooms (int)
Klassen Medical: numOfBeds (int)
Båda klasserna skall överskugga (override) den abstrakta metoden **getBuildingInfo**. Metoden skall returnera en String; en text, värdet på den valfria instansvariabeln samt arean på byggnaden. Exempel på output finns sist i uppgiften.
Klasserna behöver inte vara stora utan de skall bara vara så pass kompletta att de tillsammans med Building och IBuilding kan kompileras. Små syntax-fel överses vid rättning om de inte påverkar arkitekturen för lösningen.

Uppgift 10d: Dynamisk bindning

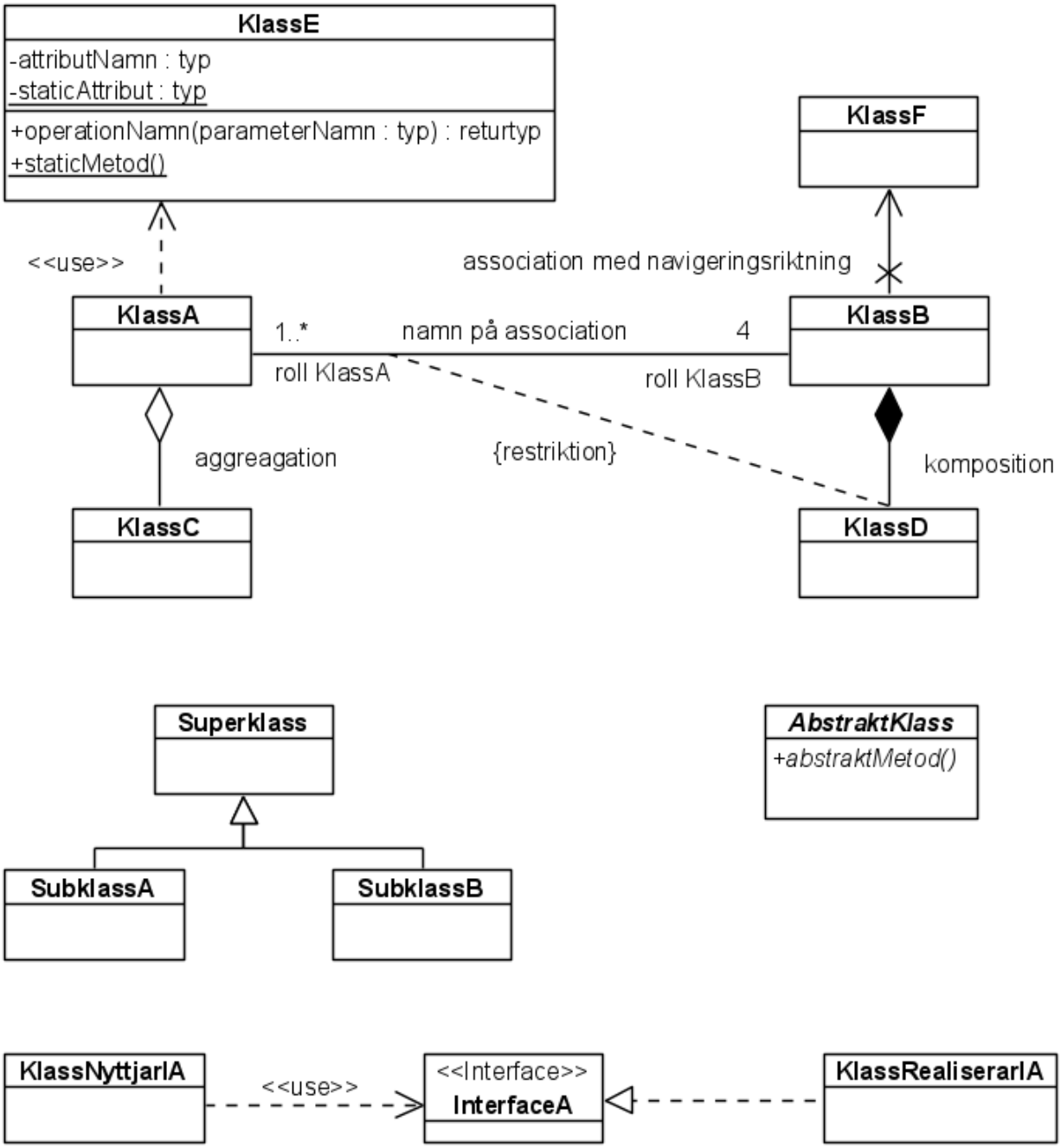
Skriv färdigt metoden createBuilding (se nedan) så main-metodens anrop fungerar för följande testvärden:
För Educational: numOfClassRooms = 1000
För Medical: numOfBeds = 300
För båda: l x b = 200 * 150 (längd x bredd)

```
public class MainProgram {
   public static void main(String[] args) {
      createBuilding(BuildingType.Educational);
       createBuilding(BuildingType.Medical);
   }
   public static void createBuilding(BuildingType bType) {
      //Komplettera
      //Krav:
      // - dynamisk bindning
      // - metoderna getBuildingInfo() och calculateArea()
       //   (metoden som räknar arean) måste anropas.
   }
}
```

Utdata från main-metoden:

```
Nice University
University has 1000 classrooms
Building has an area of 30000.0m2
---------------
Fantastic Hospital
Hospital has 300 beds
Building has an area of 30000.0m2
---------------
```

Bilaga 1 UML-notation klassdiagram



Bilaga 2 UML-notation sekvensdiagram