Tentamen på
Kurs:
DA339A, Objektorienterad programmering

Provkod: 2002 Tentamen 2, 2,5 hp

230210 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 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 ungefär vid 15.30 och 17.00 (tider kan variera något).

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	En klass kan endast ärva från en klass och kan endast implementera ett interface. Svar: FALSK, kan implementera många interfaces

B	Antag att metoderna method1 och method2 finns i en klass (se nedan):
	`public int[] method1 (){`
	`  /*kod*/`
	`}`
	`public void method2 (int[] numbers){`
	`  /*kod*/`
	`}`
	
	Metoderna kan användas enligt metodanropet nedan:
	`method2(method1());`
	Svar: SANT

C	Ett interface kan endast ha publika instansvariabler samt konstanter. Svar: FALSK, kan inte ha instansvariabler utan bara konstanter
D	En abstrakt metod måste implementeras av alla subklasser till superklassen som innehåller den abstrakta metoden. Svar: SANT
E	Om en klass är abstrakt kan man inte skapa objekt av klassen. Svar: SANT
F	Ett interface innehåller vanligtvis inte implementationen av metoder. Svar: SANT
G	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
H	Antag att klassen C2 är en subklass till klassen C1. C2 klassens egna konstruktor kommer anropas och exekveras alltid först och sen exekveras C1 klassens konstruktor. Svar: FALSK

I	Antag att klassen SavingsAccount är en subklass till klassen Account. Klassen SavingsAccount har en metod addStuff.
	Kan man skriva:
	`Account account = new SavingsAccount();`
	`account.addStuff();`
	Svar: FALSK, måste skriva ((SavingsAccount)account).addStuff();

J	Alla metoder i ett interface är implicit `public` och `abstract`. Svar: SANT
K	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
L	Överlagring (overloading) är ett annat namn för överskuggning (overriding), dvs.båda används vid arv. Svar: FALSK
M	Default accessmodifierare för klassmedlemmar (instansvariabler och metoder) är `protected` som ger åtkomst till alla metoder i samma paket och subklasser i andra paket. Svar: SANT
N	Ett sekvensdiagram är ett statiskt diagram. Svar: FALSKT, är ett dynamiskt diagram
O	En use-association är en svagare association mellan två klasser än en vanlig enkel association. Svar: SANT
P	Om man tillämpar mönstret Model-View-Controller strikt ska klasser av typen Model kommunicera direkt med klasser av typen View Svar: FALSKT
Q	En klass med stereotypen boundary kan vara en klass som kommunicerar med ett annat digitalt system. Svar: SANT
R	En klass B kan inte vara superklass till en annan klass C och subklass till en tredje klass A samtidigt. Svar: FALSKT
S	En array som skickas som argument till en metod fungerar precis som att primitiva typer, int, double, etc. vilket innebär att ev. ändringar i metoden kommer inte att påverka den ursprungliga arrayen i den anropande metoden. Svar: FALSKT
T	Vid generalisering fyller multiplicitet inget syfte i ett klassdiagram. Svar: SANT

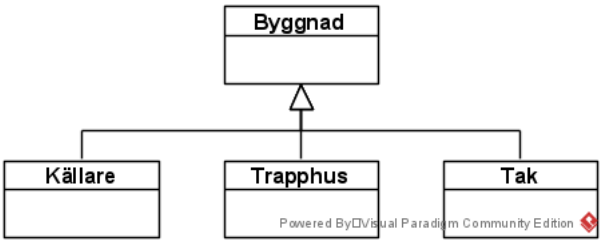
Uppgift 2 2p

Ge ett (1) exempel på en lämplig komposition och ett (1) exempel på en lämplig aggregation och använd dessa exempel för att kortfattat förklara skillnaden mellan en komposition och en aggregation.

Kommentar för svar:
Exempel som ges behöver vara rimliga/bra exempel på helhet som byggs upp av delar för att illustrera grunden i vad aggregation/komposition innebär. Förklaringen i skillnad behöver innehålla att en komposition illustrerar ett starkare förhållande mellan helheten och delarna där delarna endast kan ingå i en helhet än i en aggregation där delarna kan ingå i flera olika helheter.

Uppgift 3 2p

Diagrammet nedan ska illustrera ett program som används för att konstruera byggnader och olika element i byggnader. Motivera huruvida den givna generaliseringen nedan är lämplig eller inte.



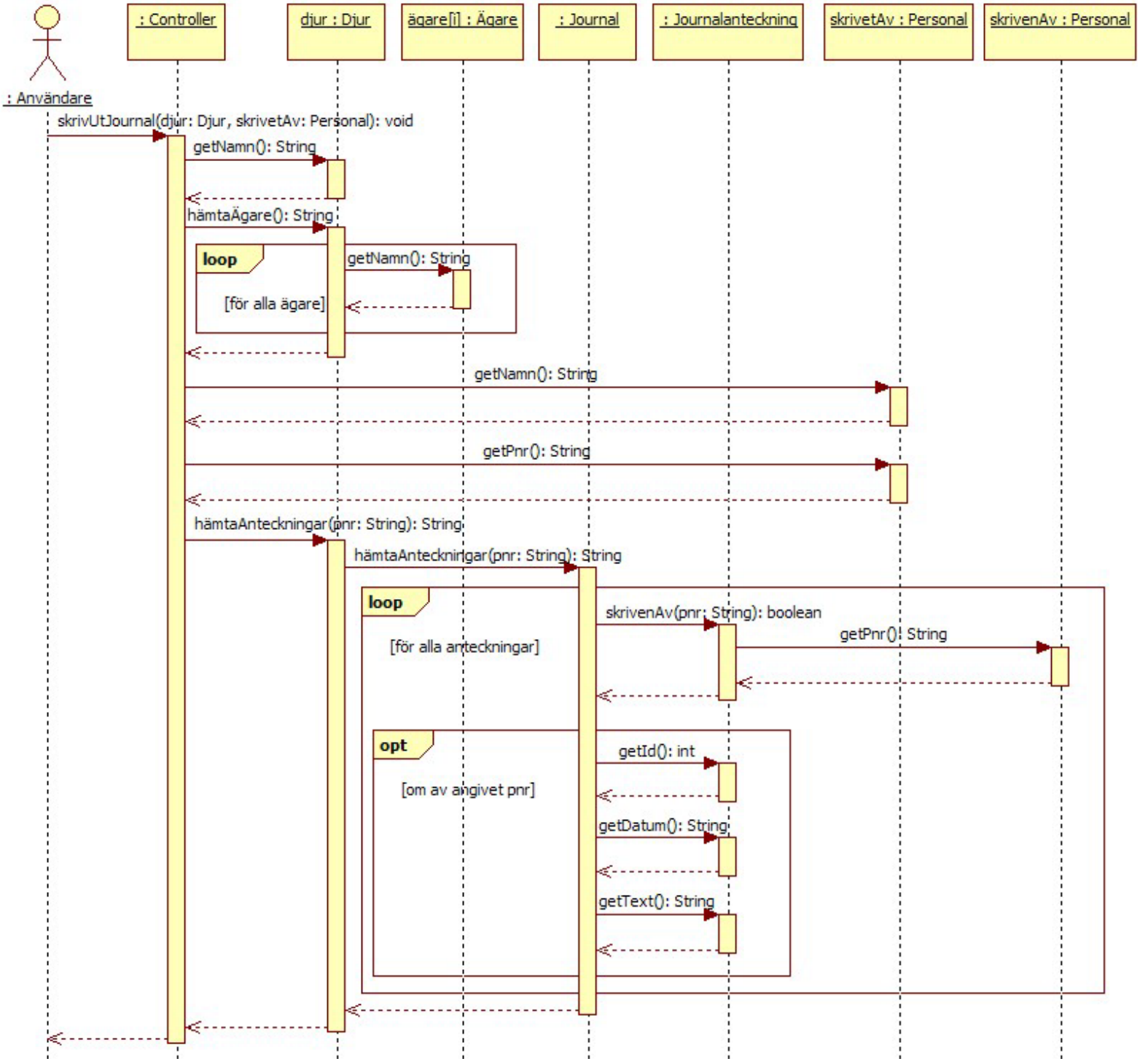
Lösningsförslag:
Nej, det är inte en lämplig generalisering eftersom en källare, trapphus eller tak inte är en hel byggnad och därför inte kan sägas vara specialiseringar av en byggnad. De är snarare delar av en byggnad. I det här fallet så hade en komposition varit lämpligare.

Uppgift 4 4p

På nästa sida finns en bild med ett sekvensdiagram. I sekvensdiagrammet finns information om klasser i form av klassnamn och metoder i dessa klasser samt viss information om vad som sker i metoderna vid körning av programmet. Skriv den kod som går att utläsa ur diagrammet för klasserna Djur och Journal.

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

Lägg inte till metoder eller annan kod som inte alls kan utläsas från diagrammet. Det är dock godtaget att göra rimliga antaganden om koden givet informationen i diagrammet (exempelvis villkor för iterationer och selektioner).



Lösningsförslag

```
class Djur {

  String getNamn(){…}

  String hämtaÄgare(){
    …
    for(i=0; i<ägare[].length; i++){
      …
      String namn = ägare[i].getNamn();
      …
    }
    …
  }

  String hämtaAnteckningar(String pnr){
    …
    String anteckningar = journal.hämtaAnteckningar(pnr);
    …
  }

}



class Journal {

  String hämtaAnteckningar(String pnr){
  …
  for(alla anteckningar){
    …
    boolean match = anteckning.skrivenAv(pnr);
    …
    if(match){
      …
      int id = anteckning.getId();
      String datum = anteckning.getDatum();
      String text = anteckning.getText();
      …
    }

}
```

Uppgift { 2p

Vad blir utskriften när metoden method() exekveras?

```
public class Test {

    public void method(double a, double b, double c, String d) {
         double sum = a+b+c;
         System.out.println("#1: result = " + d + (int)sum + "A");
     }

    public void method(int a, int b, int c, String d) {
         double sum = a+b+c;
         System.out.println("#2: result = " + (int)sum + d);
     }

    public void methodRun() {
         int a = 100;
         int b = 230;
         int c = 9;
         String d = "DA";
         method(a, b, c, d);
     }
}
```

Exempel på hur metoden kan anropas och köras:

```
public class Main {
    Test test = new Test();
    test.methodRun();
}
```

Svar:
#2: result = 339DA

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 stringTest eller inte (variabeln stringTest förutsätts vara en instansvariabel i samma klass som metoden tryToConvert).

```
String stringTest = "1,55";

public double tryToConvert(){
    double number = Double.parseDouble(stringTest);
    return number;
}
```

Lösningsförslag:

```
String stringTest = "1,55";
 //String stringTest = "1.55";
String answerToException = "Could convert: ";
 public double tryToConvert()
 {
    double number = 0;
    try
    {
       number = Double.parseDouble(stringTest);
    }
    catch (NumberFormatException e)
    {
       answerToException = "Could not convert: ";
    }
    finally
    {
       System.out.println(answerToException + stringTest + " to a
      number");
    }
    return number;
 }
```

2p: Om utskrift att lyckas efter parsing som inte exekveras om exception uppsår och catch med felmeddelande
1p: Om löst med if-satser, kan vara mindre beroende på lösning
1p:Om de fixar try – catch utan finally
1p: Om de lägger till finally och att den skriver ett meddelande om det gick att konvertera stringTest eller inte

Uppgift 7 4p

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

```
public abstract class Building {
    private String name;
    private int id;
    private String address;
    private Owner owner;

   public Building(String name, int id, String address,
                   String ownerName, String ownerPhoneNumber) {
        this.name = name;
        this.id = id;
        this.address = address;
        this.owner = new Owner(ownerName, ownerPhoneNumber);
    }

   @Override
    public String toString() {
        String textOut = String.format("Building name: %s | ID: %s |
            Address: %s | Owner: %s", name, id, address,
            owner.toString());
        return textOut;
    }
 }

public class Owner {
    private String name;
    private String phoneNumber;
    public Owner(String name, String phoneNumber){
        this.name = name;
        this.phoneNumber = phoneNumber;
    }

   public String toString() {
        String textOut = name + ", phone: " + phoneNumber;
        return textOut;
    }
 }
```

Uppgift 7a 1p

Skapa en klass University. Denna klass ska ha två instansvariabler nbrRooms med datatyp int och nbrTeachers med datatyp int. Klassen University skall ärva klassen Building (se ovan). Alla instansvariabler ska vara deklarerade med modifieraren private.

Uppgift 7b 2p

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

Uppgift 7c 1p

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

```
public class Main {
    public static void main(String[] args) {
      University university = new University("HQ", 1, "Main Street",
"Boss", "123", 80, 40);
       System.out.println(university);
    }
}
```

Output:

```
Building name: HQ | ID: 1 | Address: Main Street | Owner: Boss, phone: 123
| Rooms: 80 | Teachers: 40
```

Du kan svara på deluppgifterna i samma svar – skriva allt som efterfrågas för klassen University i ett svar. Output formatet behöver inte ha exakt samma utseende som i testkörningen ovan.

Svar/Lösningsförslag:

```
public class University extends Building {
     private int nbrRooms;
     private int nbrTeachers;
     public University(String name, int id, String address, String
ownerName, String ownerPhoneNumber, int nbrRooms, int nbrTeachers) {
          super(name, id, address, ownerName, ownerPhoneNumber);
          this.nbrRooms = nbrRooms;
          this.nbrTeachers = nbrTeachers;
     }

    //@Override
     public String toString() {
          String textOut = String.format("%s | Rooms: %s | Teachers: %s",
super.toString(), nbrRooms, nbrTeachers);
          return textOut;
     }
 }
```

Uppgift 8 7p

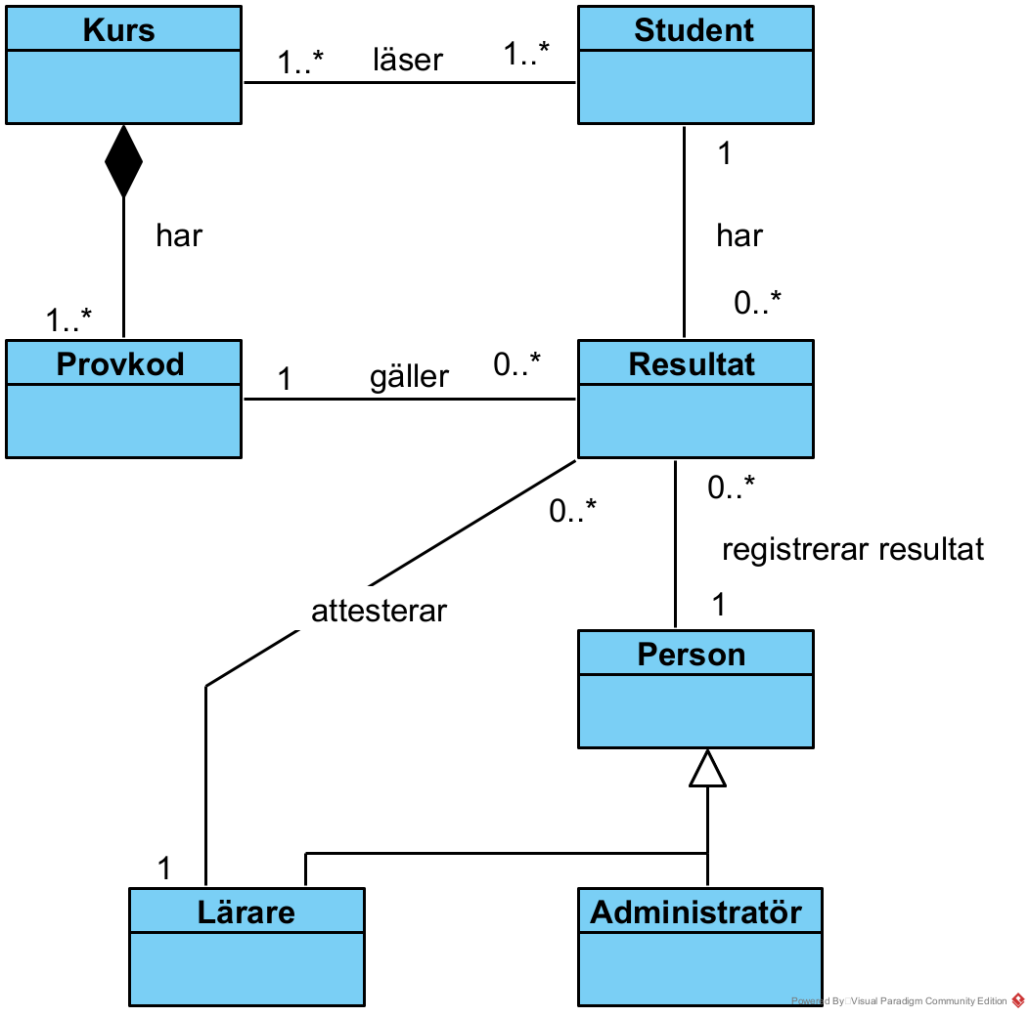
Konstruera ett lämpligt klassdiagram utifrån systembeskrivningen nedan. Identifiera lämpliga klasser för systemet utifrån beskrivningen. Använd dig av så lämpliga associationstyper som möjligt för att beskriva hur klasserna relaterar till varandra. Använd generalisering, aggregation och/eller komposition 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 system ska registrera resultat för studenter på olika kurser. En student är antagen till minst en kurs men kan ha läst eller läser flera olika kurser. Varje kurs har ett antal provkoder för vilka resultat registreras för enskilda studenter. En kurs måste ha minst en provkod. En student kan ha flera olika resultat registrerade på samma provkod. Bland personalen har olika personer olika uppgifter i förhållande till resultat som registreras för en student och en provkod. En kurslärare eller administratör kan registrera ett resultat medan en examinator attesterar ett tidigare registrerat resultat. En person kan vara både kursläraren som registrerar ett resultat och examinatorn som attesterar samma resultat. En administratör kan dock aldrig attestera resultat. Systemet behöver lagra information om vem som registrerat och attesterat ett specifikt resultat på en provkod.

Lösningsförslag:



Uppgift 9 7p

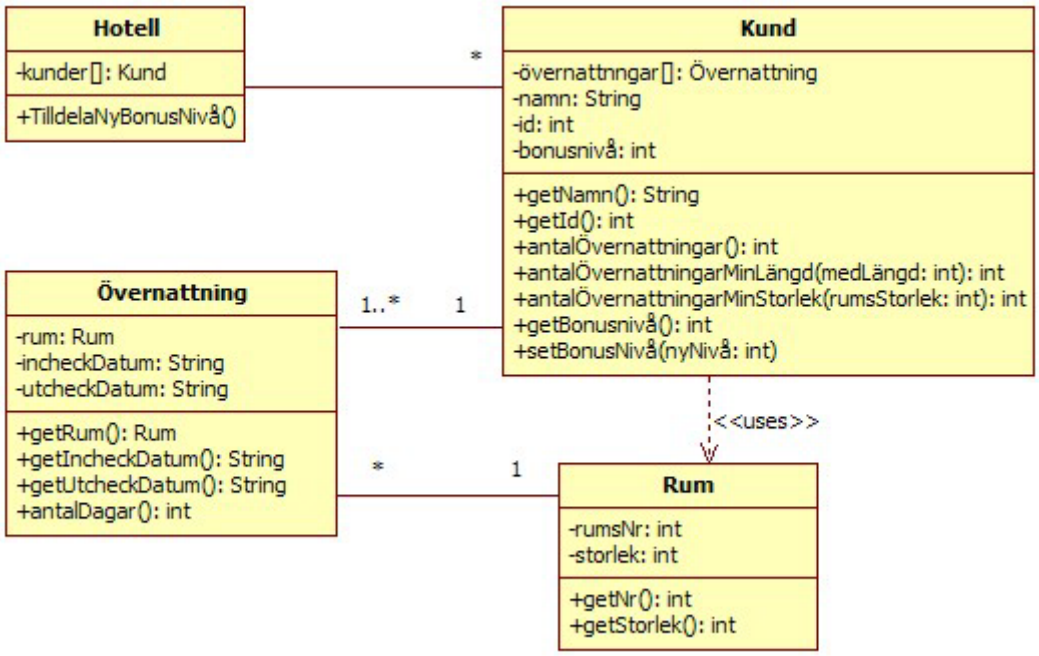
Klassdiagrammet nedan beskriver ett system som hanterar information om kunder och övernattningar för ett hotell. Givet klassdiagrammet nedan rita ett sekvensdiagram som visar vad som sker när en anställd vill köra den årliga översikten av bonusnivå för kunderna. Flödet för detta initieras via metoden TilldelaNyBounusNivå() i klassen Hotell.

Alla kunder som har gjort något av följande

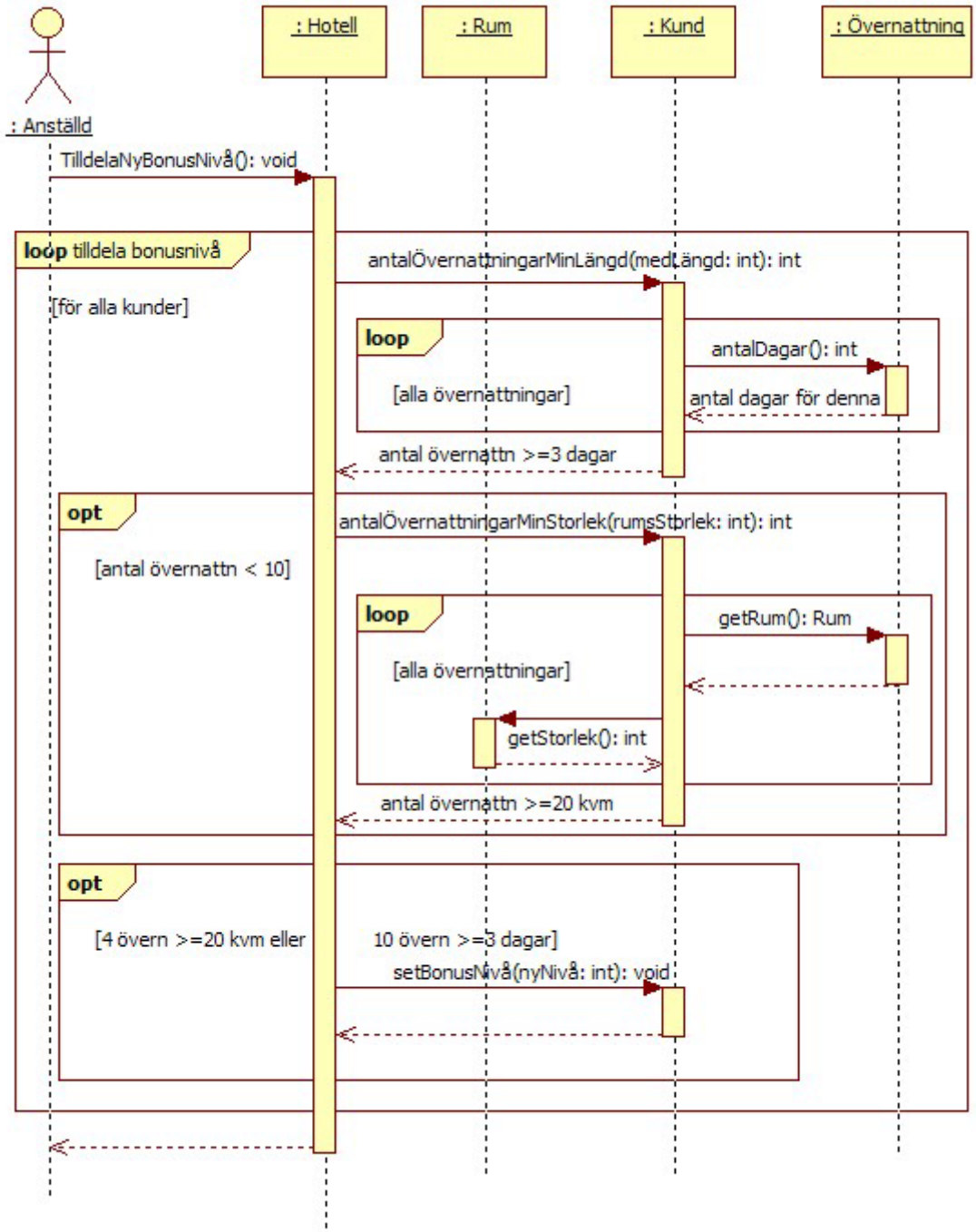
- övernattat minst 10 gånger och dessa övernattningar varat minst 3 dagar
- övernattat minst fyra gånger i ett rum som är minst 20 kvm

ska sättas till bonusnivå 3.

Du skall inte skapa några nya klasser, lägga till operationer eller attribut i befintliga klasser utan skall endast använda dig av det som visas i klassdiagrammet. Du får dock skapa en aktör som representerar den anställda på hotellet som initierar sekvensen i programmet.



Lösningsförslag



Uppgifter för betyg VG

Uppgift 10

Eumerationen **LiteratureType** är given nedan. Du ska skriva en klass **Literature** som använder enumerationen och implementerar interfacet **ILiterature**. Senare i uppgiften skall du också skriva ett par konkreta subklasser till Literature. 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 LiteratureType {
   Book,
   E_Book,
   Magazine
}
```

Uppgift 10a: Interface

Komplettera interfacet vid markeringarna 1 till 3.

```
public interface ILiterature {
   //En litteratur (literature) måste ha en typ av enum-typen
   //LiteratureType.
   //1. Komplettera med en getter- och en setter-metod som
   //   subklasserna (i detta fall klassen Literature) ska
   //   implementera.

   //En litteratur (literature) skall räkna ut och returnera antal
   //sidor som finns i litteraturen utifrån konstanten
   //nbrOfWordsPerPage och totalNbrOfWords.
   //Number of pages = totalNbrOfWords / nbrOfWordsPerPage.
   //2. Deklarera en konstant (nbrOfWordsPerPage) av datatypen int
   //   och ge konstanten värdet 250. Komplettera också med en metod
   //   som returnerar antal sidor som en int.

   //3. Skriv en metod getLiteratureInfo som skall returnera en
   //   String (mer om denna metod i uppgift 10b).
}
```

Uppgift 10b: Abstrakt klass och abstrakt metod

Skriv en klass **Literature** som implementerar interfacet. Klassen skall innehålla all kod som implementation av interfacet kräver, med undantag av metoden **getLiteratureInfo** 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).

Obs. det är **inte** obligatoriskt att skapa och använda konstruktorer i denna och de andra klasser som du skriver i denna uppgift. **Dock** om du inte använder någon konstruktor måste du använda dig av setter- och getter-metoder. Konstruktorer, setter- och getter-metoder skriver du efter behov.

Definiera **getLiteratureInfo** i klassen Literature som skall returnera en string. I denna del, skriv bara de ändringar som du behöver göra i klassen Literature.

Uppgift 10c: Konkreta klasser

Skriv två klasser, **Book** och **E_Book** som subklasser till Literature. Deklarera en valfri instansvariabel i varje klass. Ifall du inte kommer på något bra attribut, använd följande:

Klassen Book: weight (int)
Klassen E_Book: diskSpace (int)

Båda klasserna ska ha en setter-metod som ska användas när värdet på den valfria instansvariabeln sätts.

Båda klasserna skall överskugga (override) den abstrakta metoden **getLiteratureInfo**. Metoden skall returnera en String; en text, värdet på den valfria instansvariabeln samt antal sidor som litteraturen har. Exempel på output finns sist i uppgiften.

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

Uppgift 10d: Dynamisk bindning

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

E_Book: disk space = 4
Book: weight = 2
För båda: totalNbrOfWords / nbrOfWordsPerPage = 250 000 / 250 (konstanten från interfacet)

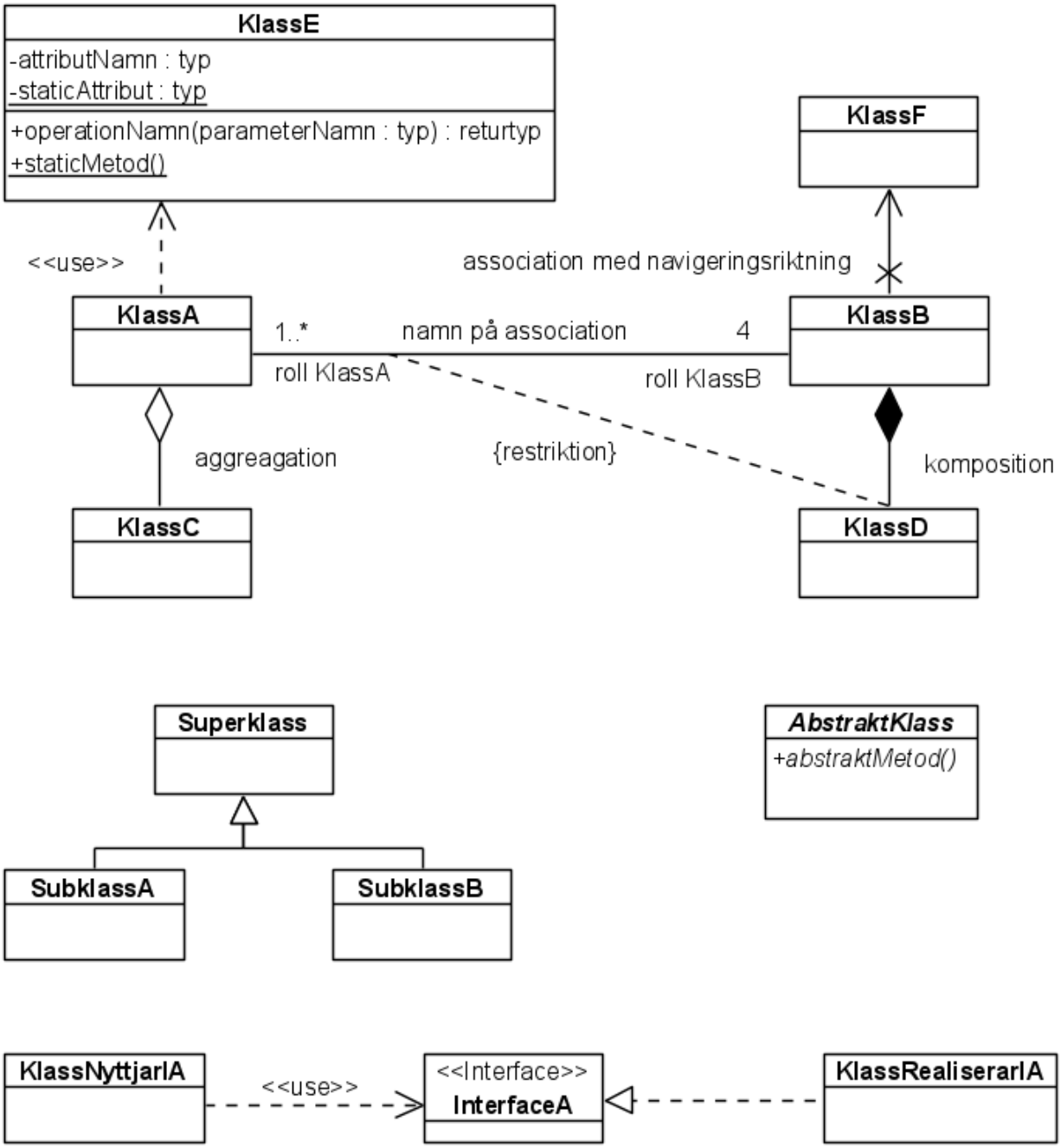
```
public class MainProgram {
   public static void main(String[] args) {
      createLiterature(LiteratureType.E_Book);
      createLiterature(LiteratureType.Book);
   }

   public static void createLiterature(LiteratureType lType) {
      //Komplettera
      //Krav:
      // - dynamisk bindning
      // - metoderna getLiteratureInfo() och
      //   calculateNumberOfPages()(metoden som räknar antal sidor)
      //   måste anropas.
   }
}
```

Utdata från main-metoden:

```
Nice E-Book
E-Book needs: 4MB of free disk space
The literature has a total of 1000 pages
---------------
Fantastic Book
Book weighs: 2kg
The literature has a total of 1000 pages
---------------
```

Bilaga 1 UML-notation klassdiagram



Bilaga 2 UML-notation sekvensdiagram