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

Provkod: 2002 Tentamen 2, 2,5 hp

230811 kl 8.15 – 13.15

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)**
- **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.

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 klassen G är en subklass till klassen T. Klassen G har en metod med namnet Boo. Kan man skriva: T t = new G(); ((G)t).Boo(); Svar: SANT
B	I en try-catch-sats med finally-sats, kan man bara ha ett try-block, bara ett finally-block, och bara ett catch-block. Svar: FALSK kan ha ett eller flera catch-block
C	Man måste alltid definiera metoden toString i en klass för att kunna använda den. Svar: FALSK
D	En klass kan ärva från flera klasser och kan implementera flera interfacer. Svar: FALSK, kan implementera många interfaces
E	Man kan skapa ett objekt av en klass som är abstrakt. Svar: FALSK

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

G	En enum är en klasstyp och kan instansieras, dvs. det går att skapa ett objekt av en enum med nyckelordet new. Svar: FALSK
H	En klass kan vara både en subklass och en superklass. Svar: SANT
I	Ett interface kan både implementera och ärva ett annat interface. Svar: FALSK
J	En klass som ärver från en annan klass kallas för en superklass eller basklass. Svar: FALSK
K	En abstrakt metod behöver inte implementeras av alla subklasser till superklassen som innehåller den abstrakta metoden. Svar: FALSK
L	Ett interface innehåller vanligtvis implementation av metoder.

	Svar: FALSK
M	En superklass konstruktor anropas och exekveras alltid först och sen exekveras subklassens egna konstruktor. Svar: SANT
N	Navigeringsriktning på en association visar vilken klass som anropar den andra klassen i associationen. Svar: SANT
O	Vid generalisering ska multiplicitet alltid sättas ut. Svar: FALSKT
P	Om två klasser har flera associationer mellan sig får dessa inte vara av samma typ. Svar: FALSKT
Q	Ett objekt har alltid en unik identitet. Svar: SANT
R	Ett sekvensdiagram är ett statiskt diagram. Svar:FALSKT
S	Om man tillämpar konceptet Boundary-Control-Entity strikt så ska klasser av typen Boundary inte kommunicera direkt med klasser av typen Entity. Svar: SANT
T	En klass kan ha en association till sig själv. Svar: SANT

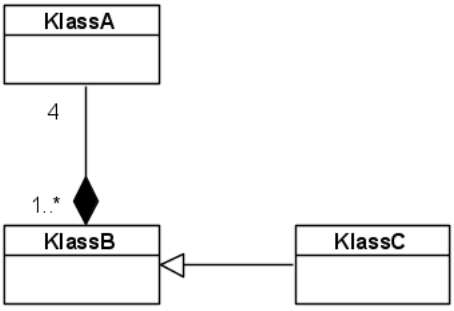
Uppgift 2 2p

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.

Exempel på 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 är de delar som 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 , exempelvis förändra View utan att behöva förändra Model.

Uppgift 3 2p

Diagrammet nedan kan innehålla viss felaktig och/eller orimlig information i fråga om de klasser och associationer som finns. Förklara eventuella problem som finns i diagrammet nedan gällande klasser och associationer och varför det är ett problem.



Svar

I en komposition kan inte delen, KlassA, tillhöra flera helheter, KlassB, vilket nu är fallet. Om ett objekt av KlassB raderas i en komposition så ska alla delar det vill säga objekt av KlassA också raderas. Om delarna då hör till flera helheter så blir det problem i koden då vad som sker i ett objekt påverkar ett annat helhets-objekts länkar till del-objekt.

Uppgift 4 2p

Vad blir utskriften när metoden sumLetters exekveras?

```
public class Overloading {
   public String letters(String a, String b){
       String text = ("#111: text = " + a + b + "a");
       return text;
   }

   public String letters (char a, char b){
       String text = ("#222: text = " + a + b + "c");
       return text;
   }

   public void sumLetters(){
       char a = 'a';
       char b = 'b';
       System.out.println(letters(a, b));
   }
}
```

Exempel på hur metoden kan anropas och köras:

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

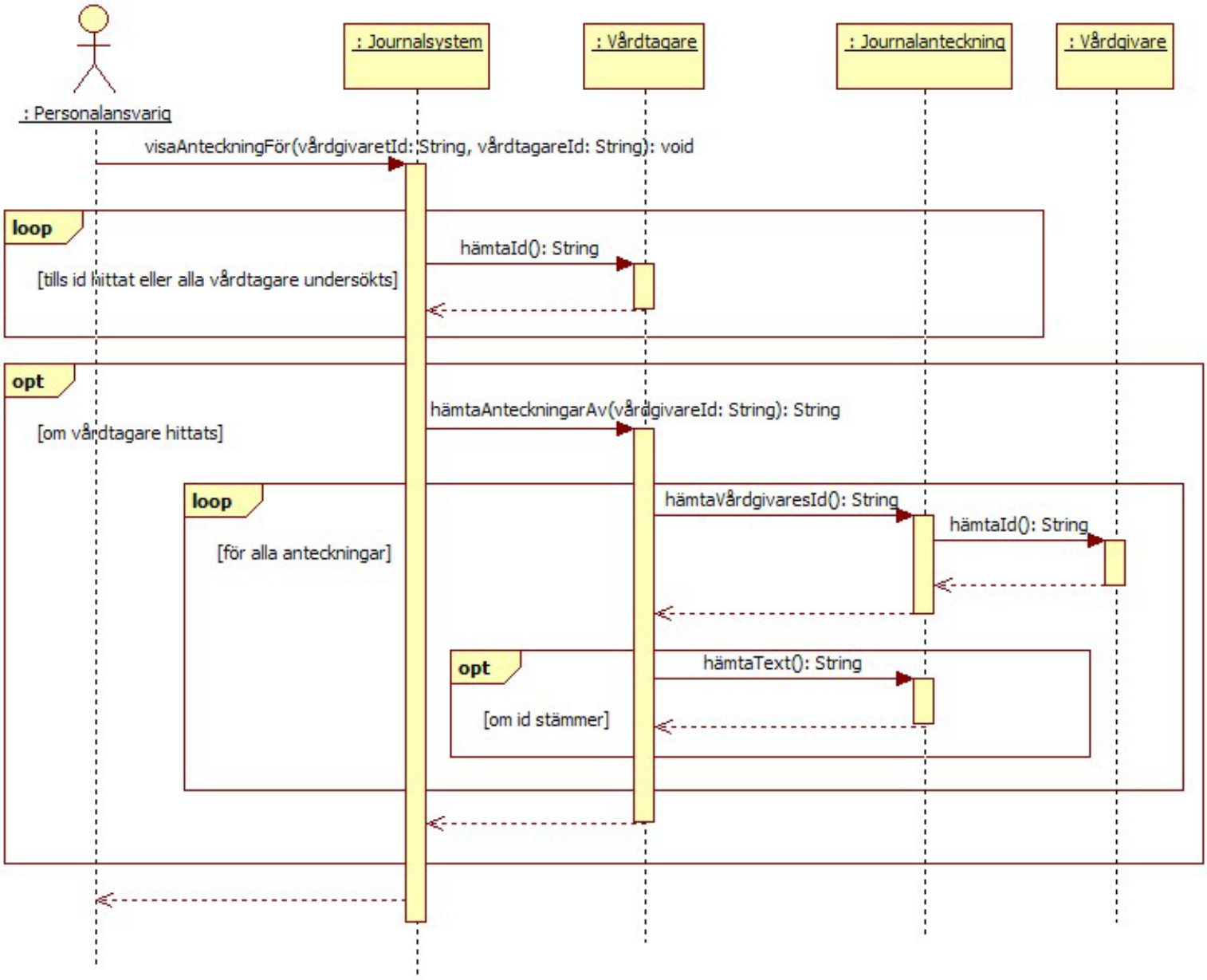
Svar/Lösningsförslag:
#222: text = abc

Uppgift 5 4p

I sekvensdiagrammet nedan finns information om klassnamn och metoder i dessa klasser samt viss information om vad som sker i metoderna. Skriv kod som går att utläsa ur diagrammet för de två klasserna:

- Vårdtagare
- Journalanteckning

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 namn på lokala variabler, villkor i iterationer och selektioner samt villkor i dessa.



Lösningsförslag

```
class Vårdtagare{
  String hämtaId{
    …
    return id;
  }

  String hämtaAntackningarAv(String vårdgivareId){
    …
    while(flerAntecknignar){
      aktuellId = aktuellAnteckning.hämtaVårdgivaresId();
      if (aktuellID == vårdgivareId){
        text = aktuellAnteckning.hämtatext();
      }
    }
  }
}

class Journalanteckning{
  String hämtaVårdgivaresId(){
    …
    return vårdgivare.hämtaID();
  }

  String hämtaText(){
    …
    return text;
  }
}
```

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 = "123ABC";

public int convertToNumber()
{
   int number = Integer.parseInt(stringNumber);
   return number;
}
```

Lösningsförslag:

```
String stringNumber = "123ABC";
 //String stringNumber = "339";
String answerToException = "Could convert: ";
 public int convertToNumber()
 {
    int number = 0;
    try
    {
       number = Integer.parseInt(stringNumber);
    }
    catch (NumberFormatException e)
    {
       answerToException = "Could not convert: ";
    }
    finally
    {
       System.out.println(answerToException + stringNumber + " to a
       number");
    }
    return number;
 }
```

2p:
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 stringNumber eller inte
2p: Om utskrift att lyckas efter parsing som inte exekveras om exception uppsår och catch med felmeddelande
Max 1p om löst med if-satser, kan vara mindre beroende på lösning

Uppgift 7a-c 4p

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

```
public abstract class Vehicle {
   private String name;
   private int created;
   private Configuration config;

   public Vehicle(String name, int created, String brand,
                  String model){
      this.name = name;
      this.created = created;
      this.config = new Configuration(brand, model);
   }

   public String toString() {
       String textOut = String.format("Name: %s | Created: %s |
         Configuration: %s", name, created, config.toString());
       return textOut;
    }
}

public class Configuration {
   private String brand;
   private String model;

   public Address(String brand, String model){
       this.brand = brand;
       this.model = model;
   }

   public String toString(){
       String textOut = brand + " " + model;
       return textOut;
   }
}
```

Uppgift 7a 1p

Skapa en klass Car. Denna klass ska ha två instansvariabler color med datatyp String och carID med datatyp String. Klassen Car skall ärva klassen Vehicle (se ovan). Alla instansvariabler ska vara deklarerade med modifieraren private.

Uppgift 7b 2p

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

Uppgift 7c 1p

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

```
public class Main {
    public static void main(String[] args) {
       Car car = new Car("Top Speed", 23, "Brand X",
                         "ABC123", "Blue", "DA339A");
       System.out.println(car);
    }
}
```

Output:

```
Name: Top Speed | Created: 23 | Configuration: Brand X, ABC123 | Color:
Blue | Car ID: DA339A
```

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

Svar/Lösningsförslag:

```
public class Car extends Vehicle {
    private String color;
    private String carID;
    public Car(String name, int created, String brand, String model, String
color, String carID)
    {
        super(name, created, brand, model);
        this.color = color;
        this.carID = carID;
     }

    //@Override
    public String toString()
    {
     String textOut = String.format("%s | Color: %s | Car ID: %s",
super.toString(), color, carID);
       return textOut;
    }
}
```

Uppgift 8 7p

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 Bilaga 1).

Systembeskrivning

På en veterinärklinik måste man föra journaler över de djur man behandlar. För varje djur som behandlas 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 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 görs detta genom att man registrerar namn och kontaktuppgifter.

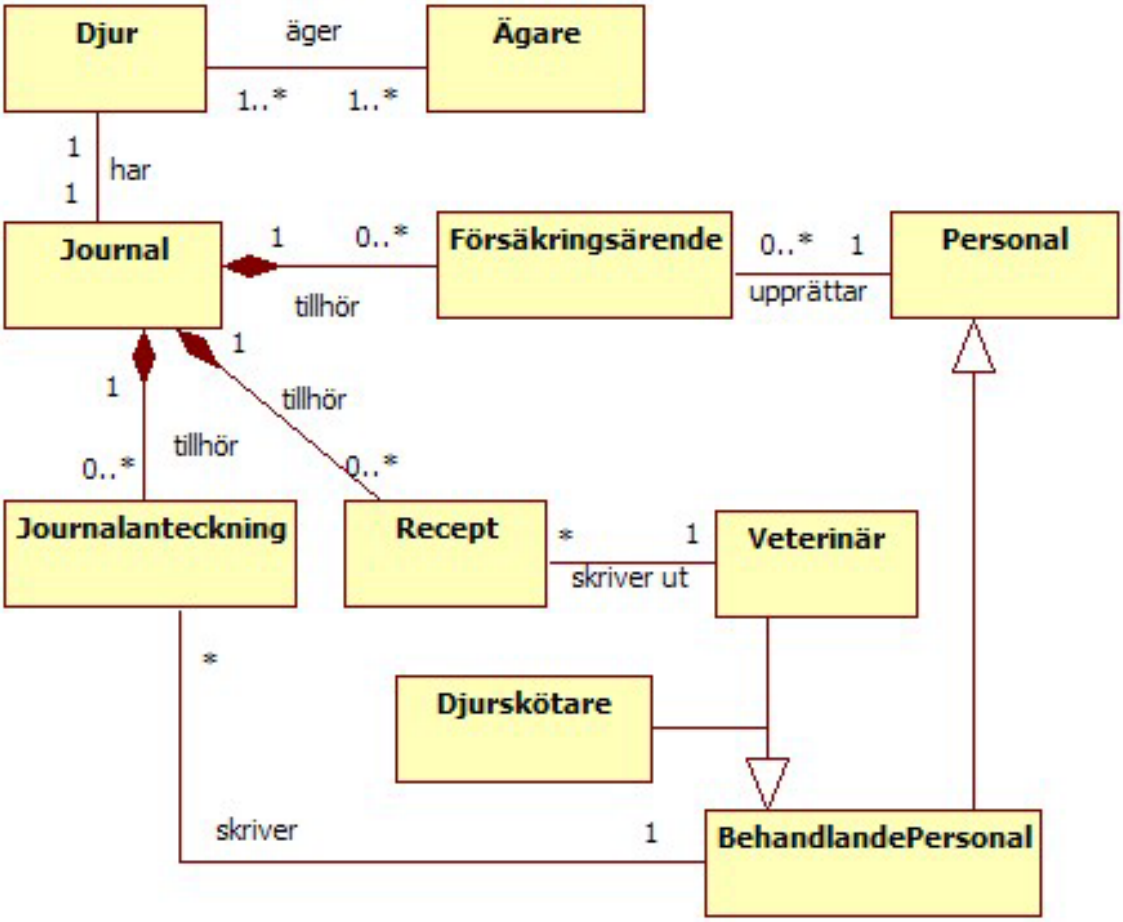
För att underlätta för ägare som har sina djur försäkrade hos något försäkringsbolag så sköter veterinärkliniken viss kontakt med försäkringsbolagen genom att skicka in uppgifter som krävs för att ägaren ska få ersättning på sin försäkring. Till journalen kan då även finans kopplat ett antal försäkringsärenden.

På veterinärkliniken arbetar olika typer av personal som kan behöva arbeta med journalsystemet.

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 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

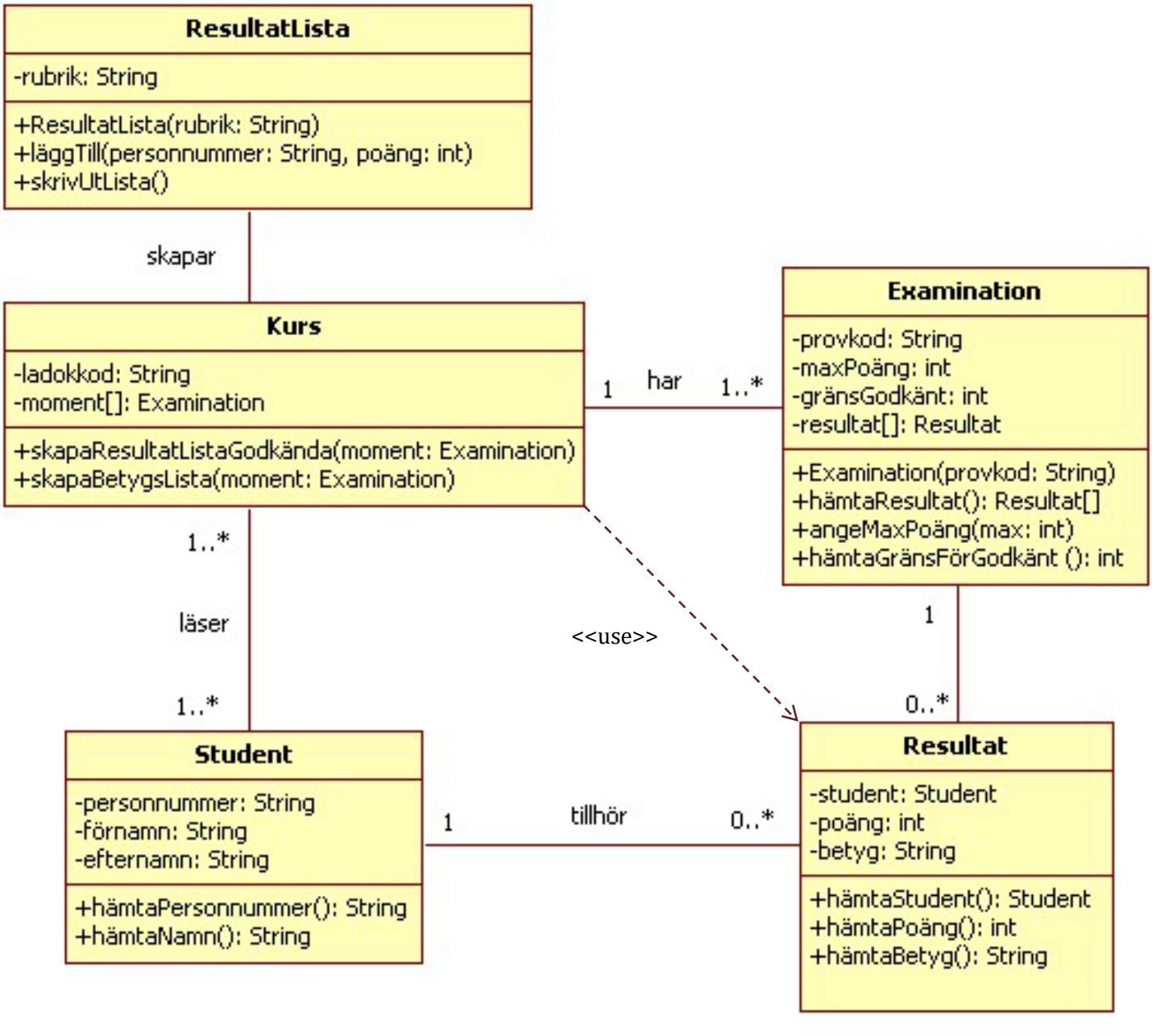


Uppgift 9 7p

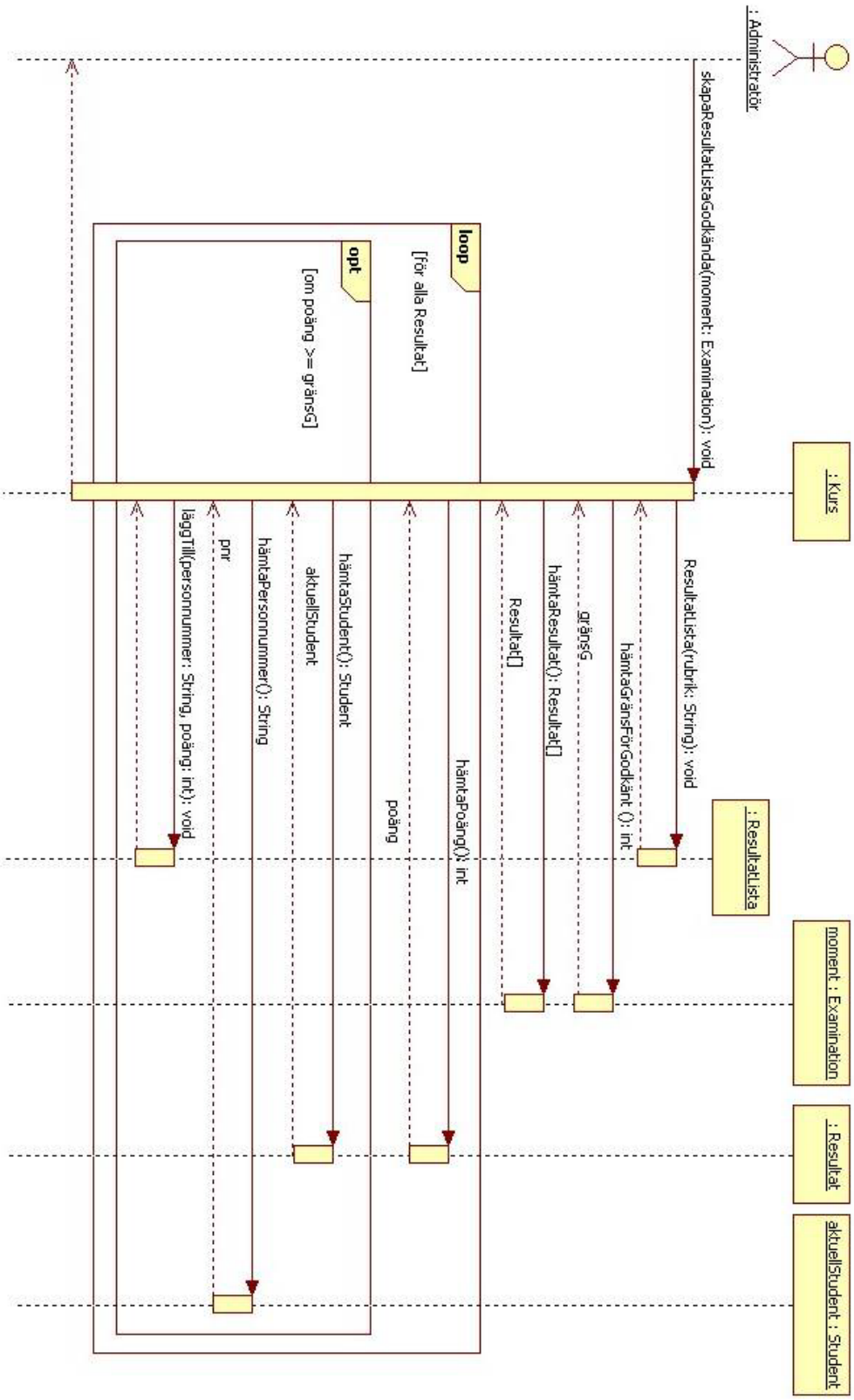
Klassdiagrammet nedan beskriver ett system som hanterar studieresultat vid ett lärosäte.

Givet klassdiagrammet nedan rita ett sekvensdiagram som visar vad som sker när en administratör initierar till ett objekt av klassen Kurs att skapa en lista över alla godkända resultat för ett visst angivet examinationsmoment (metoden skapaResultatListaGodkända(monet: Examination)). Listan skall visa personnummer och poäng för de resultat som är godkända. Det vill säga inga resultat med poäng mindre än gränsen för godkänt skall läggas till i listan.

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 dock skapa en aktör som representerar en administratör i diagrammet om behov finns.



Lösningsförslag



Uppgifter för betyg VG

Uppgift 10a-d

Eumerationen **BuildingBlockType** är given nedan. Du ska skriva en klass **BuildingBlock** som använder enumerationen och implementerar interfacet **IBuildingBlock**. Senare i uppgiften skall du också skriva ett par konkreta subklasser till BuildingBlock. 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 BuildingBlockType {

   Cube,

   Triangle,

   Cylinder

}
```

Uppgift 10a: Interface

Komplettera interfacet vid markeringarna 1 till 3.

```
public interface IBuildingBlock {

   //En byggkloss (BuildingBlock) måste ha en typ av enum-typen
BuildingBlockType.

   //1. Komplettera med en getter- och en setter-metod som
   //   subklasserna (i detta fall klassen BuildingBlock) ska
   //   implementera.

   //En byggkloss (BuildingBlock) skall räkna ut och returnera sin
   //volym (volume) utifrån sin längd, höjd, bredd samt typ av
byggkloss.

   //- Om det är av typen Cube: Volym = bredd*höjd*längd

   //- Om det är av typen Triangle: Volym = (bredd*höjd*längd)/2

   //2. Komplettera med metod som returnerar volymen som en double.
   //3. 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 **BuildingBlock** som implementerar interfacet ovan. Klassen skall innehålla all kod som implementation av interfacet kräver, med undantag av metoden **getBuildingBlockInfo** 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 **getBuildingBlockInfo** i klassen BuildingBlock som skall returnera en string.

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

Uppgift 10c: Konkreta klasser

Skriv två klasser, **Cube** och **Triangle** som subklasser till BuildingBlock. 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)

Båda klasserna skall överskugga (override) den abstrakta metoden **getBuildingBlockInfo**. Metoden skall returnera en String; en text, värdet på den valfria instansvariabeln samt volymen på byggklossen. Exempel på output finns sist i uppgiften.

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

Uppgift 10d: Dynamisk bindning

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

Cube: num of cube sides = 6
Triangle: num of triangle sides = 5

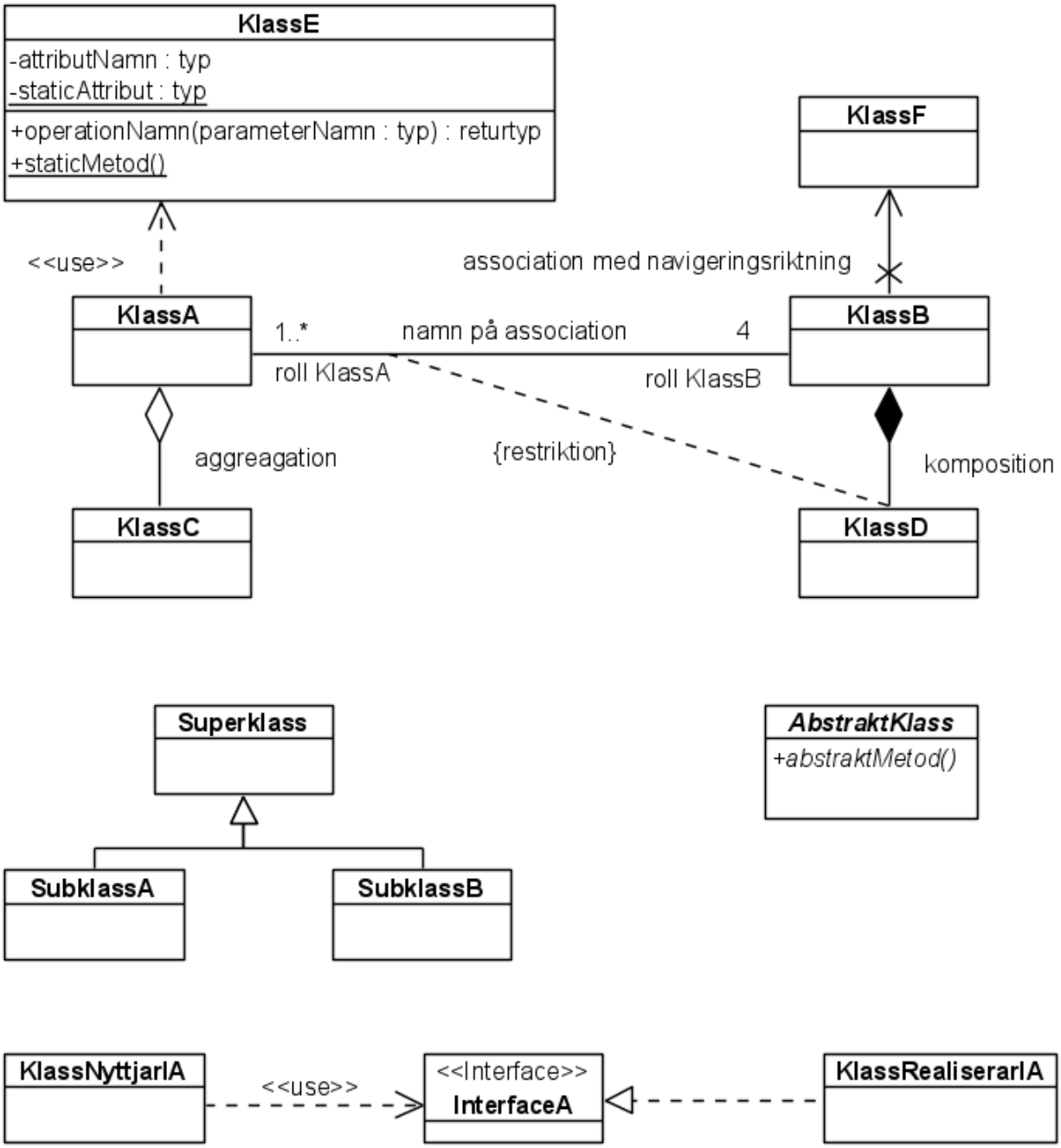
För Cube: b x h x l = 3 * 5 * 8 (bredd x höjd x längd)
För Triangle: (b x h x l) / 2 = (3 * 5 * 8) / 2 ((bredd x höjd x längd) / 2)

```
public class MainProgram {

   public static void main(String[] args) {

      createBuildingBlock(BuildingBlockType.Cube);
      createBuildingBlock(BuildingBlockType.Triangle);

   }

   public static void createBuildingBlock(BuildingBlockType bType) {

      //Komplettera

      //Krav:
      // - dynamisk bindning
      // - metoderna getBuildingBlockInfo() och
calculateBlockVolume(BuildingBlockType bType)
      //   (metoden som räknar volymen, och får in vilken typ av
byggkloss det är) måste anropas.

   }

}
```

Utdata från main-metoden:

```
Nice cube

cube has 6 sides

Building block has a volume of 120.0m3

---------------

Fantastic triangle

triangle has 5 sides

Building block has a volume of 60.0m3

---------------
```

Bilaga 1 UML-notation klassdiagram



Bilaga 2 UML-notation sekvensdiagram