Tentamen på
Kurs:
DA339A, Objektorienterad programmering

Provkod: 2002 Tentamen 2, 2,5 hp

230105 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)**

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 vid start och ungefär vid 9.30 och 11.00 (tider kan variera något då det är många som tentar och tentan kan vara uppdelad på flera salar).

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 setStudentNames (String[] names){`
	`  /*kod*/`
	`}`
	Kan metoden anropas från en annan metod så här:
	`setStudentNames(studNames[]);`
	Svar: FALSK

B	Ett interface kan ha publika och privata instansvariabler samt konstanter. Svar: FALSK
C	Nyckelordet `implements` används för att implementera ett interface och nyckelordet `extends` används för att ärva en superklass. Svar: SANT
D	En abstrakt metod måste implementeras av alla subklasser till superklassen som innehåller den abstrakta metoden. Svar: SANT förutsätter att subklasserna är konkreta klasser då ingen annan info finns
E	En subklass ärver alla publika medlemmar av sin superklass inklusive konstruktor(er). Svar: FALSK
F	Alla klasser i Java är direkt eller indirekt en subklass till klassen Object. Svar: SANT
G	En klass B kan vara superklass till en annan klass C och subklass till en tredje klass A samtidigt. Svar: SANT
H	En enum är en klasstyp och kan instansieras, dvs. det går att skapa ett objekt av en enum med nyckelordet new. Svar: FALSK
I	Om vi inte definierar metoden toString i en klass, finns metoden ändå tillgänglig. Svar: SANT
J	Ett interface kan varken ärva från en klass eller implementera ett annat interface. Svar: SANT – kan inte implementera ett interface – ett interface kan inte *implementera* något och ett interface kan inte ärva en klass (men kan utöka ett annat interface)
K	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
L	Antag att klassen B är en subklass till klassen A. Klassen B har en metod Foo. Kan man skriva:

	`A a = new B();`
	`a.Foo();`
	Svar: FALSK, måste skriva ((B)a).Foo();

M	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
N	En klass kan inte ha en association till sig själv. Svar: FALSKT
O	Ett klassdiagram visar vilka objekt som finns när programmet körs. Svar: FALSKT klassdiagram visar endast möjliga klasser och är ett statiskt diagram som visar strukturen på koden – inte vad som sker under körning
P	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
Q	Navigeringsriktning på en association visar vilken klass som anropar den andra klassen i associationen. Svar: SANT
R	Överlagring (overloading) innebär att man kan ha flera metoder samma namn men med olika uppsättning av parametrar. Vid överskuggning (overriding) har man en och samma metod i superklassen men med olika implementation i olika subklasser i en klasshierarki. Svar: SANT
S	När ett 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: FALSKT det sker en form av pass-by-reference som skickar adressen till objektet, objektets data kopieras inte
T	Objekt kan dela identitet och klass med varandra. Svar: FALSKT objekt kan dela klass men har alltid skilda identiteter

Uppgift 2 2p

Förklara begreppet polymorfism i förhållande till begreppet generalisering.

Svar: Se beskrivningar i F17 och senare av begreppet polymorfism
Beskrivningar av generalisering i sig självt ger inte poäng
När vi säger att en objekt kan ha flera former som förklaring på polymorfism avses flera former för ETT objekt, inte flera olika subklasser till en superklass eller olika varianter av metoder i form av override eller overload.

Uppgift 3 2p

Förklara en (1) nackdel och en (1) fördel med att använda en strikt MVC-arkitektur (Model-View-Controller) i ett program.

Svar: Se material förknippat med F16
Saker som gäller generellt för objektorientering ger inte poäng. Det måste vara specifikt för MVC-mönstret (exempelvis att varje klass har ett tydligt syfte eller ansvarsområde - detta är inte specifikt för MVC utan för OO generellt). Den personliga upplevelsen av att något kan vara svårt eller komplext är inte heller specifikt för mönstret och ger inte poäng.

Uppgift 4 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 Forum och Tråd.

Koden skall visa vilka metoder klasserna Forum och Tråd 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 någon kod som inte kan utläsas från diagrammet eller rimligt göra antagande om givet informationen i diagrammet (exempelvis villkor för iterationer och selektioner).

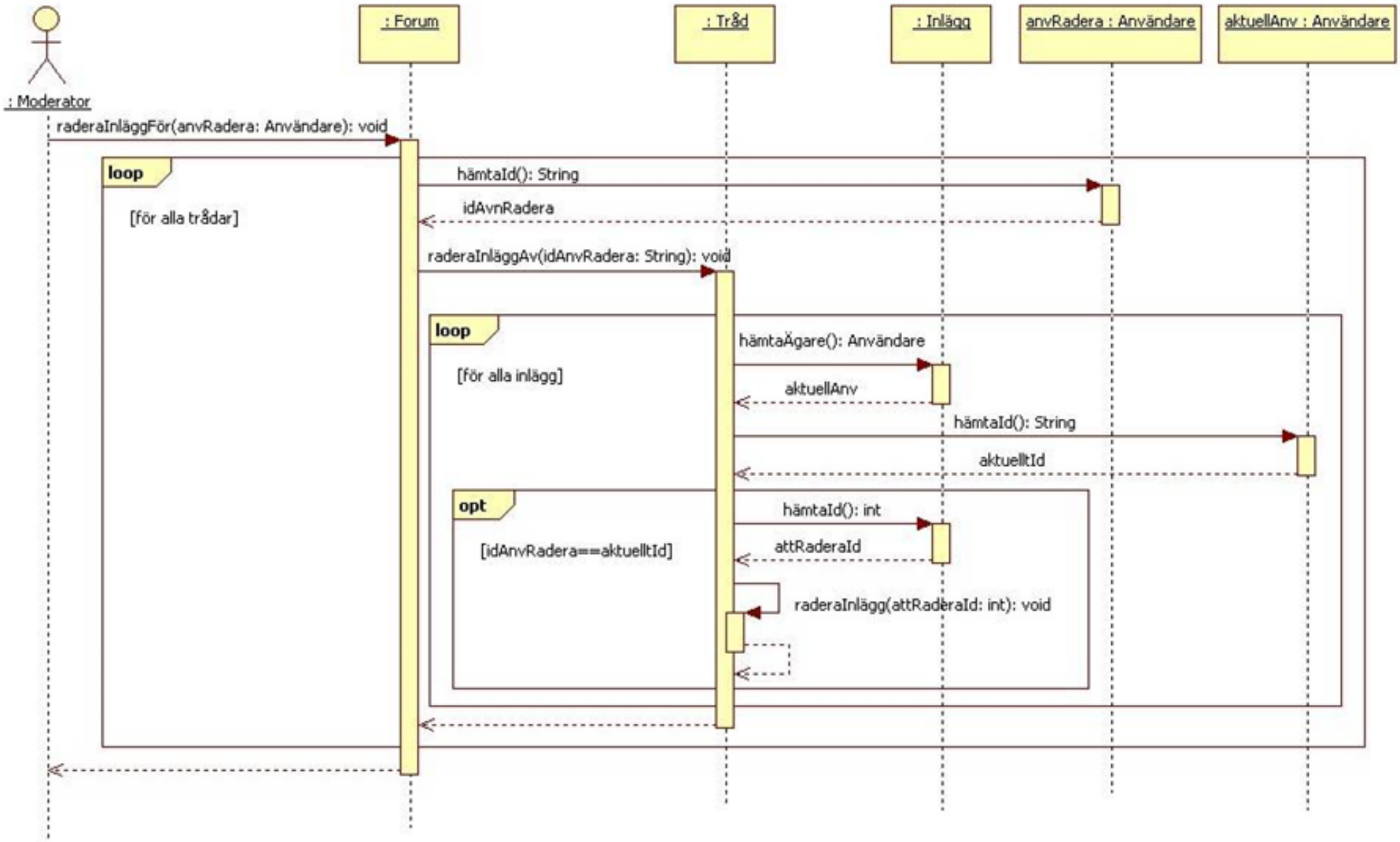
Lösningsförslag

```
class Forum {
  void raderaInläggFör(Användare anvRadera){
    for(alla trådar){
      String idAnvRadera = anvRadera.hämtaId();
      aktuellTråd.raderaInläggAv(idAnvRadera);
    }
  }
}

class Tråd{
  void raderaInläggAv(String idAnvRadera){
    for(alla inlägg){
      Användare aktuellAnv = aktuelltInlägg.hämtaÄgare();
      String aktuelltId = aktuellAnv.hämtaId();

      if(aktuelltId==idAnvRadera){
        int attRaderaId = aktuelltInlägg.hämtaId();
        raderaInlägg(attRaderaId);
      }
    }
  }

  void raderaInlägg(int attRaderaId){…}
}
```


: Moderator
: Forum
: Tråd
: Inlägg
anvRadera : Användare
aktuellAnv : Användare
raderaInläggFör(anvRadera: Användare): void
loop
[för alla trådar]
hämtaId(): String
idAvnRadera
raderaInläggAv(idAnvRadera: String): void
loop
[för alla inlägg]
hämtaÄgare(): Användare
aktuellAnv
hämtaId(): String
aktuelltId
opt
[idAnvRadera==aktuelltId]
hämtaId(): int
attRaderaId
raderaInlägg(attRaderaId: int): void

Uppgift 5 2p

Vad blir utskriften när metoden sumLetters exekveras?

```
public class TestOverloading {

   public String letters(String a, String b)
   {
      String text = ("#1: text = " + a + b + "a");
      System.out.println(text);
      return text;
   }

   public String letters (char a, char b)
   {
      String text = ("#2: text = " + a + b + "c");
      System.out.println(text);
      return text;
   }

   public void sumLetters()
   {
      String a = "a";
      String b = "b";
      letters(a, b);
   }
}
```

Exempel på hur metoden kan anropas och köras:

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

Svar:
#1: text = aba

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 = "DA339A";

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

Lösningsförslag:

```
String stringNumber = "DA339A";
//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: 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 stringNumber eller inte

Uppgift 7 4p

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

```
public abstract class Person {
   private String name;
   private int age;
   private Address address;

   public Person(String name, int age, String city, String street){
      this.name = name;
      this.age = age;
      this.address = new Address(city, street);
   }

   public String toString() {
      String textOut = String.format("Namn: %s | Ålder: %s år |
          Address: %s", name, age, address.toString());
      return textOut;
   }
}

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

   public Address(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 Teacher. Denna klass ska ha två instansvariabler title med datatyp String och courseID med datatyp String. Klassen Teacher skall ärva klassen Person (se ovan). Alla instansvariabler ska vara deklarerade med modifieraren private.

Uppgift 7b 2p

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

Uppgift 7c 1p

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

```
public class Main {
   public static void main(String[] args) {
      Teacher teacher = new Teacher("Sebastian", 34, "Falsterbo",
          "Sandvipe street", "University lecturer", "DA339A");
      System.out.println(teacher);
   }
}
```

Output:

```
Name: Sebastian | Age: 34 | Address: Falsterbo, Sandvipe street | Title:
University lecturer | Course ID: DA339A
```

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

Lösningsförslag

```
public class Teacher extends Person{
    private String title;
    private String courseID;

    public Teacher(String name, int age, String city, String street,
   String title, String courseID)
   {
       super(name, age, city, street);
       this.title = title;
       this.courseID = courseID;
    }

   @Override
    public String toString()
   {
       String textOut = String.format("%s | Title: %s | Course ID:
      %s", super.toString(), title, courseID);
       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).

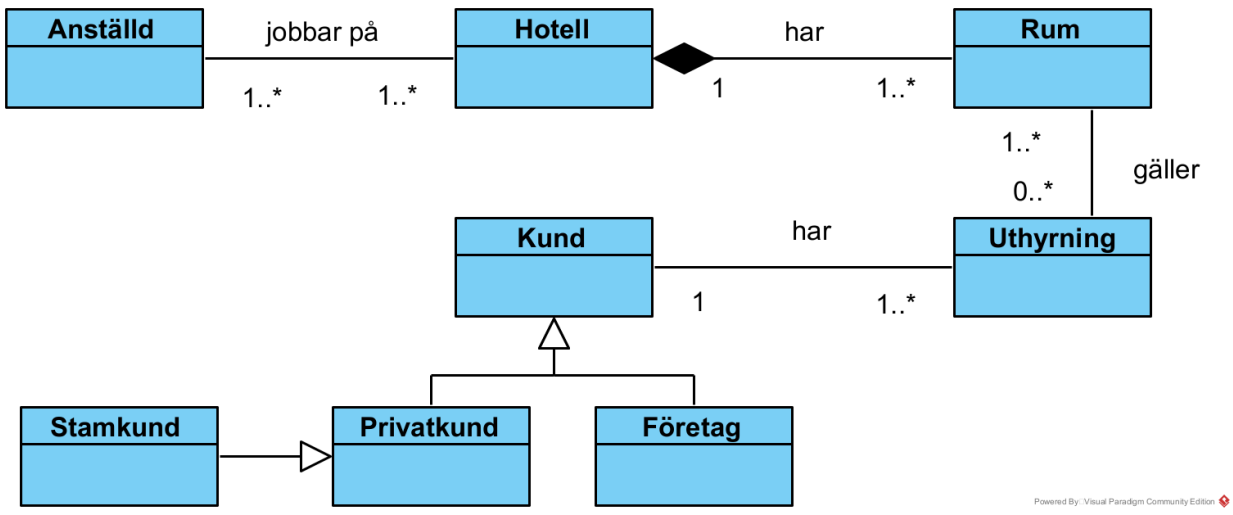
Systembeskrivning
Systemet ska administrera en hotellkedjas bokningar för kunder på de hotell man har i kedjan samt vilka anställda som arbetar på vilka hotell.

Varje hotell har ett antal rum som kan hyras ut till kunder. En kund kan hyra flera rum åt gången och man vill även hålla reda på historik för vilka rum en kund tidigare hyrt. För en uthyrning behöver man veta start och slutdatum för uthyrningen för att veta om ett rum är ledigt eller inte vid ett visst tillfälle eller när det var uthyrt till en viss kund.

Kunder kan vara privatpersoner eller företag som hyr rum åt sin personal. Privatpersoner kan vara stamkunder vilket innebär att man har extra fördelar om man ofta hyr rum hos hotellkedjan. Företag kan inte vara stamkunder utan avtal med förmåner hanteras separat för varje företag.

Anställda kan arbeta på flera olika hotell i hotellkedjan. Man vill veta vilka hotell de anställda jobbar på för att veta vilka rutiner de är familjära med då dessa kan skifta mellan olika hotell.

Lösningsförslag:



Uppgift 9 7p

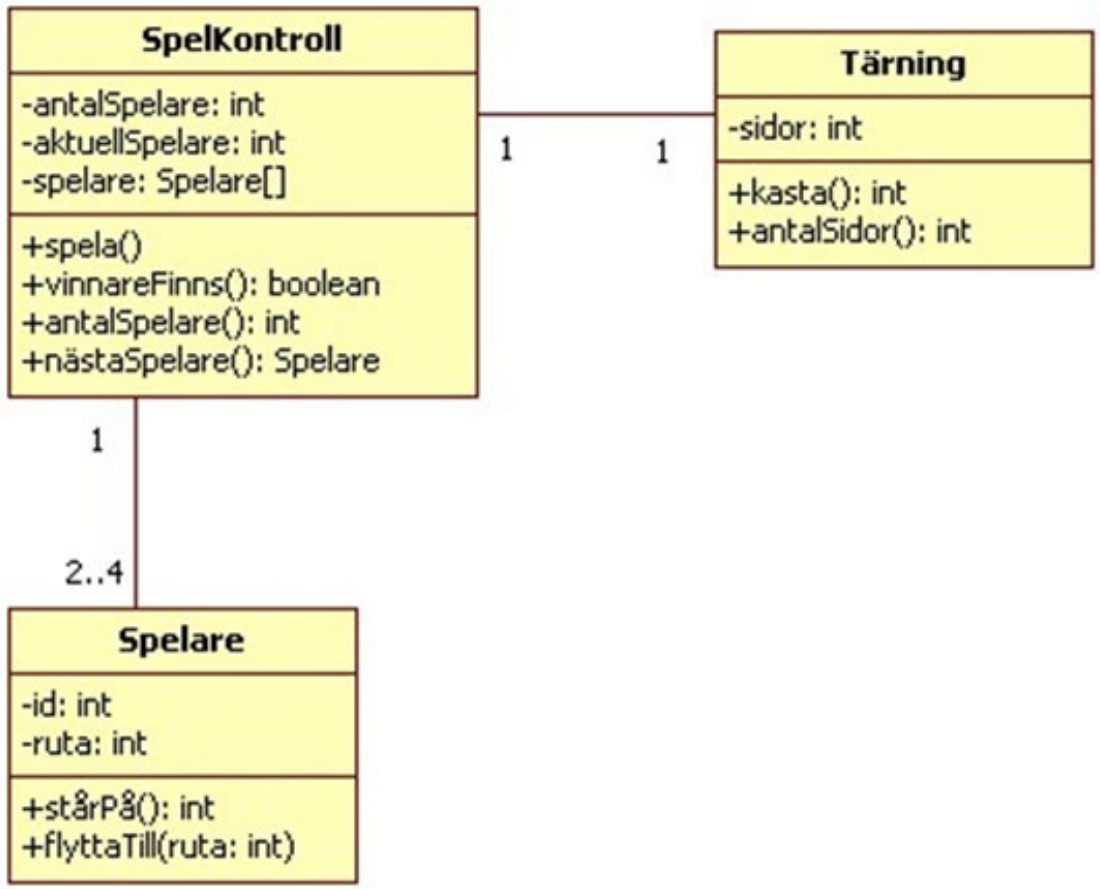
Klassdiagrammet nedan beskriver ett system som simulerar ett brädspel. Rita ett sekvensdiagram som visar vad som sker när ett spel simuleras med följande process: Simuleringen går igenom spelarna i en bestämd ordning tills en vinnare uppstår. I varje omgång sker följande för den aktuella spelaren: en tiosidig tärning slås och beroende på resultatet av tärningskastet flyttar spelaren enligt följande regler:

- 1-6 – motsvarande antal steg framåt
- 7-8 – ett steg bakåt
- 9-10 – står still

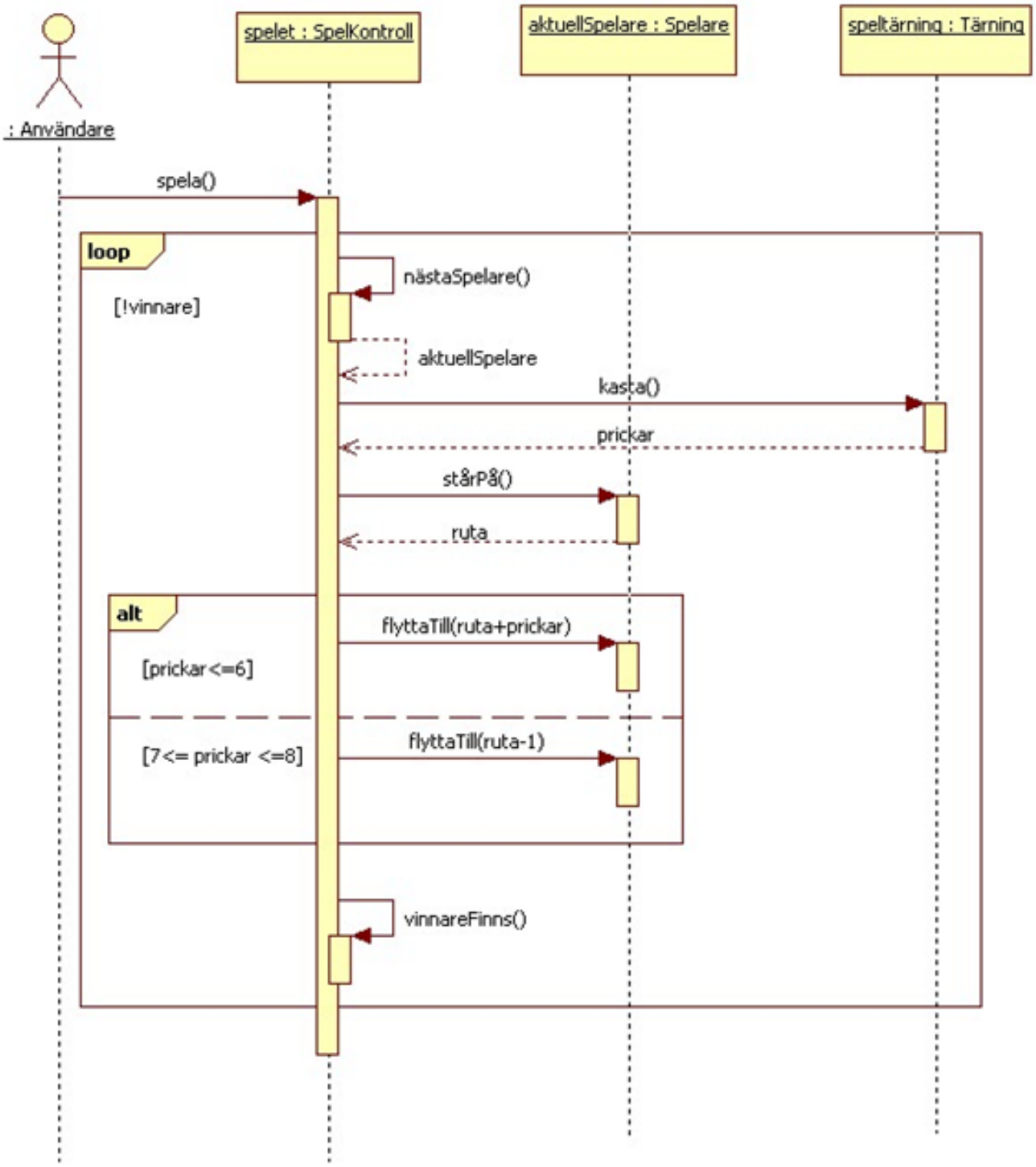
Den spelare som först slutför ett varv runt spelplanen vinner. Spelplanen har 100 rutor numrerade 1-100. Simuleringen av spelet startas med ett anrop till metoden spela() i klassen SpelKontroll.

Rita ett sekvensdiagram som visar simuleringen av spelet. Du kan anta att antalet spelare är bestämt och alla objekt redan är skapade.

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 användaren som startar simuleringen genom att anropa operationen spela().



Lösningsförslag:



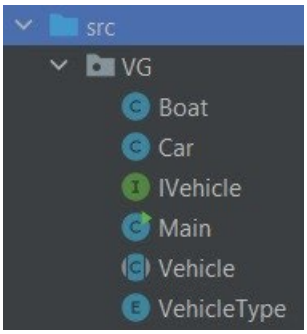
Uppgifter för betyg VG

Uppgift 10

Eumerationen **VehicleType** är given nedan. Du ska skriva en klass **Vehicle** som använder enumerationen och implementerar interfacet **IVehicle**. Senare i uppgiften skall du också skriva ett par konkreta subklasser till Vehicle. 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 VehicleType {
    Car,
    Boat,
    Plane
}
```



Uppgift 10a: Interface

Komplettera interfacet vid markeringarna 1 till 3.

```
public interface IVehicle {
    //Ett fordon (vehicle) måste ha en typav enum-typen VehicleType.
    //1. Komplettera med en getter- och en setter-metod som
    //   subklasserna (i detta fall klassen Vehicle) ska
    //   implementera.

    //Ett fordon (Vehicle) skall räkna ut och returnera sin
    //lastkapacitet (cargo capacity) utifrån sin längd, höjd och
    //bredd. Volym = bredd * höjd * längd.
    //2. Komplettera med metod som returnerar volymen som en double.

    //3. Skriv en metod getVehicleInfo som skall returnera en String.
    //   (mer om denna metod i uppgift 8b).
}
```

Uppgift 10b: Abstrakt klass och abstrakt metod

Skriv en klass **Vehicle** som implementerar interfacet. Klassen skall innehålla all kod som implementation av interfacet kräver, med undantag av metoden **getVehicleInfo** 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. Setter- och getter-metoder skriver du efter behov.

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

Uppgift 10c: Konkreta klasser

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

Klassen Car: numOfWheels (int)
Klassen Boat: numOfPropeller (int)

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

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

Uppgift 10d: Dynamisk bindning

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

Car: num of wheels = 8
Boat: num of propellers = 4
För båda: b x h x l = 3 * 5 * 8 (bredd x höjd x längd)

```
public class MainProgram {
   public static void main(String[] args) {
      createVehicle(VehicleType.Car);
      createVehicle(VehicleType.Boat);
   }

   public static void createVehicle(VehicleType vType) {
      //Komplettera
      //Krav:
      // - dynamisk bindning
      // - metoderna getVehicleInfo() och calculateCargoCapacity()
      //   (metoden som räknar volymen) måste anropas.
   }
}
```

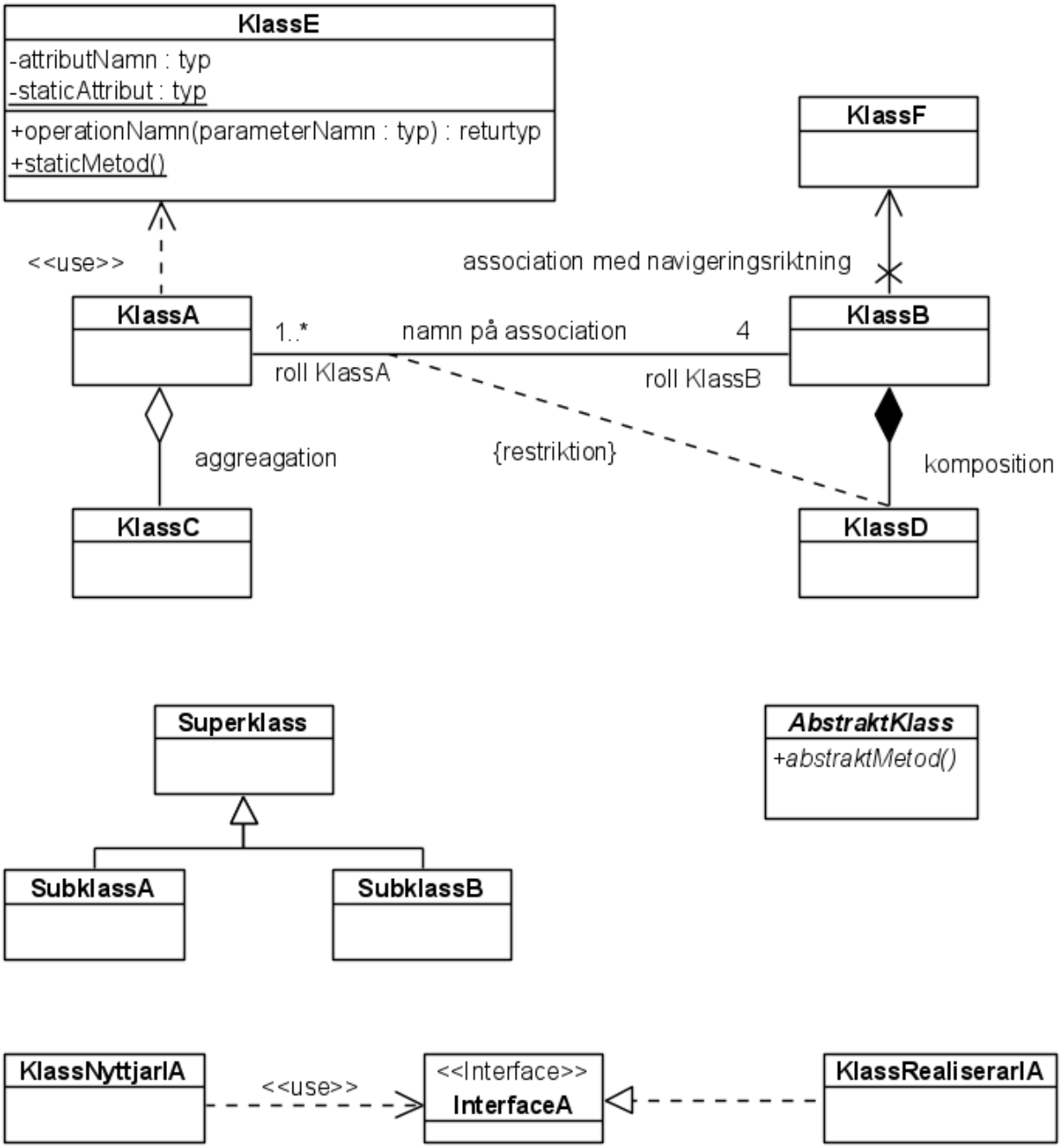
Utdata från main-metoden:

```
Nice car
car has 8 wheels
Vehicle has a cargo capacity of 120.0m3
---------------
Fantastic boat
boat has 4 propellers
Vehicle has a cargo capacity of 120.0m3
---------------
```

Lösningsförslag:

Finns separat zip-fil (*Tentamen2DA339AHT22VGSolution.zip*) med kod.

Bilaga 1 UML-notation klassdiagram



Bilaga 2 UML-notation sekvensdiagram