Tentamen på
Kurs:
DA339A, Objektorienterad programmering

Provkod: 2002 Tentamen 2, 2,5 hp

210105 kl 08.15 – 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: 1-2 uppgifter, bedöms som godkända eller inte godkända

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 bägge uppgifterna på del med uppgifter för betyg VG bedöms som godkända

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? 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”. Om det bara är två listor med bokstäver vet vi inte vilka svar som du anser är vad.

Fråga	Påstående
A	Objekt kan dela identitet och klass med varandra.
B	Vid generalisering fyller multiplicitet inget syfte i ett klassdiagram.
C	Det kan endast finnas en association mellan två klasser.
D	Om man tillämpar mönstret Model-View-Controller strikt så ska klasser av typen Model kommunicera direkt med klasser av typen View.
E	Det går inte att skapa ett objekt av en abstrakt klass.
F	Sekvensdiagram är en dynamisk typ av diagram.
G	Vid hantering av undantag, kan det finnas endast ett **try** och ett **finally** block men flera **catch** block.
H	Om ett program försöker spara en fil till en enhet som är skrivskyddad kastas ett "Checked" undantag oavsett om koden hanterar undantaget eller inte.
I	Alla klasser i Java ärver från klassen **Object** direkt om klassen **inte är** en subklass till en annan klass och indirekt om klassen **är** en subklass till någon annan klass.
J	Ett interface kan ha publika och privata instansvariabler samt konstanter.
K	Ett interface kan varken ärva från en klass eller implementera ett annat interface.
L	En klass kan endast implementera ett interface och ärva från endast en klass.
M	Vid överlagring (overloading) sker bindning (dvs. vilken metod skall anropas) statiskt vid kompilering.
N	Nyckelordet **throw** används inne i en metod och följs av en undantagstyp och kastar alltid undantaget.
O	När en klass definierar en eller flera abstrakta metoder, måste klassen definieras som abstract. En abstract metod måste definieras med nyckelordet abstract.
P	Nyckelordet **throws** används tillsammans en metods signatur och därmed slipper man att använda try-catch inne i metoden.
Q	Skillnaden mellan en abstract-metod och en överskuggad metod är en abstrakt metod har implementation (kropp) i superklassen men en överskuggad metod behöver inte ha en kropp i subklassen.
R	En abstrakt metod måste implementeras av alla subklasser till superklassen som innehåller den abstrakta metoden..

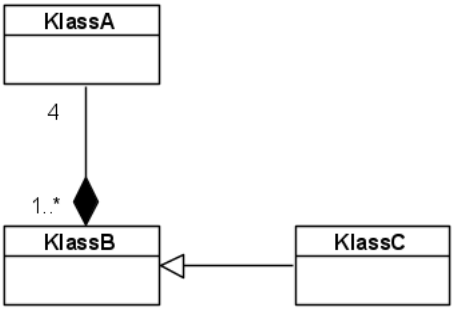
S	Default accessmodifierare för klassmedlemmar (instansvariabler och metoder) är protected som ger åtkomst till alla metoder i samma paket och subklasser i andra paket.
T	En klass som ärver från en annan klass kallas för en subklass, eller härledd (derived) klass.

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

Fråga	Påstående
A	Falskt – objekt kan dela klass med varandra men identitet är unikt för varje objekt
B	Sant
C	Falskt – det kan finnas flera associationer och av olika former mellan klasser. Se exempel i F13.
D	Falskt – I en strikt MVC-struktur så ska model och view inte kommunicera direkt med varandra.
E	Sant – den abstrakta klassen har inte all kod som krävs för att skapa ett objekt.
F	Sant – Sekvensdiagram visar hur något sker – vad händer i koden när den körs.
G	Sant
H	Falskt
I	Sant
J	Falskt
K	Falskt
L	Falskt
M	Sant
N	Sant
O	Sant
P	Sant
Q	Falskt
R	Sant
S	Sant
T	Sant

Uppgift 2 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 3 2p

Vi kan dela in klasser i de olika typerna/stereotyperna boundary, control och entity. Förklara kortfattat vad som skiljer de olika typerna åt och vad som utmärker respektive typ av klass.

Svar: Se beskrivningar av de olika typerna i F15

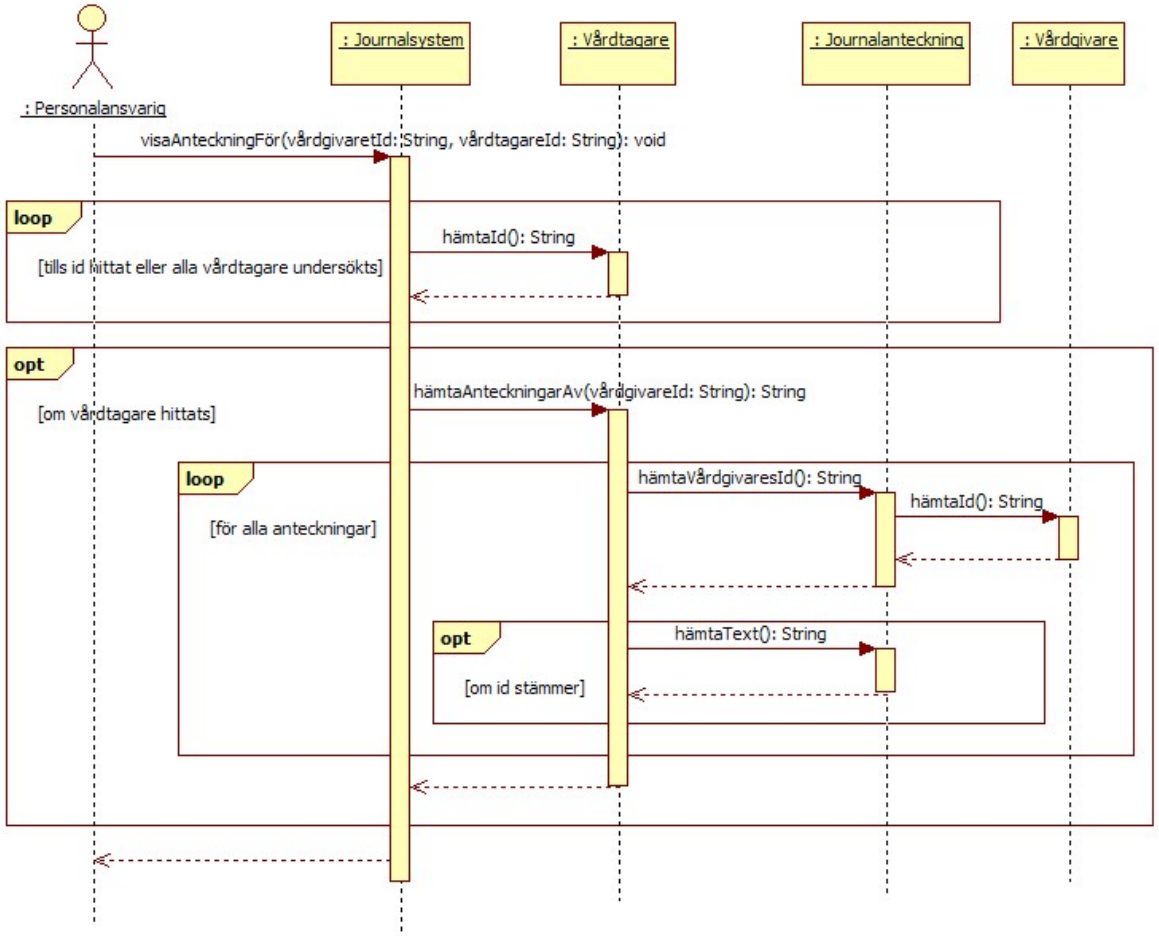
Uppgift 4 4p

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

- Vårdtagare
- Journalanteckning

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 Vårdtagare{
  String hämtaId{
    …
    return id;
  }

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

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

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

Uppgift 5a 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.

```
String strTal = "abc100";
 public double convertToTal()
 {
     double tal = Double.parseDouble(strTal);
     return tal;
 }
```

svar:

```
String strTal = "abc100";
public double convertToTal()
{
    double tal = 0.0;
    try
    {
        tal = Double.parseDouble(strTal);
    }
    catch (NumberFormatException e)
    {

    }
    return tal;
}
```

Går också bra med Exception i stället för NumberFormatException

Uppgift 5b 3p

Koden nedan är given.

```
class Bike
{
    private int numOfGears;
    private String model;

    public Bike(String model, int gears)
    {
        //kod
    }
}
```

Skriv en klass **ElBike** med en instansvariabel **motorWatt** (double). Skriv inga andra instansvariabler och ingen metod. Skriv en konstruktor som kan anropas med värden för att initiera model, numOfGears och motorWatt. Ett objekt av ElBike skall kunna skapas som t.ex i satsen nedan:

```
Bike bike = new ElBike("Mountain Ranger", 18, 36.0);
```

Svar:

```
class ElBike extends Bike
{
    double motorWatt;
    public ElBike( String model, int gears,  double watt)
    {
        super(model, gears);
        this.motorWatt = watt;

    }
    public String toString()
    {
        String text = String.format("%s och %.2f W motor.",
super.toString(), motorWatt);
        return text;
    }

    public static void main(String[] args)
    {
        ElBike kidsBike = new ElBike("Mountain Ranger", 18, 36.0);
        System.out.println(kidsBike.toString());
    }
}
```

Uppgift 5c 3p

Skriv kod i konstruktorn till klassen **Bike** så in-parametrarna sparas i objektet. Komplettera också med en **toString** metod i både **Bike** och **ElBike**. Metoden i **Elbike** skall anropa Bike:s **toString** så vi kan testa klasserna så här:

```
public static void main(String[] args)
{
    ElBike kidsBike = new ElBike("Mountain Ranger", 18, 36.0);
    System.out.println(kidsBike.toString());
}
```

Och en testkörning med de ovanstående värden skall ge följande utdata:

```
Mountain Ranger, 18 växlar och 36,00 W motor.
```

Svar se 5b

Uppgift 6 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 lämpligt. Skriv ut namn på alla associationer för att förtydliga dessa. Om det går att ange en multiplicitet så skall detta finnas. Gör inga tillägg med information som inte finns i systembeskrivningen.

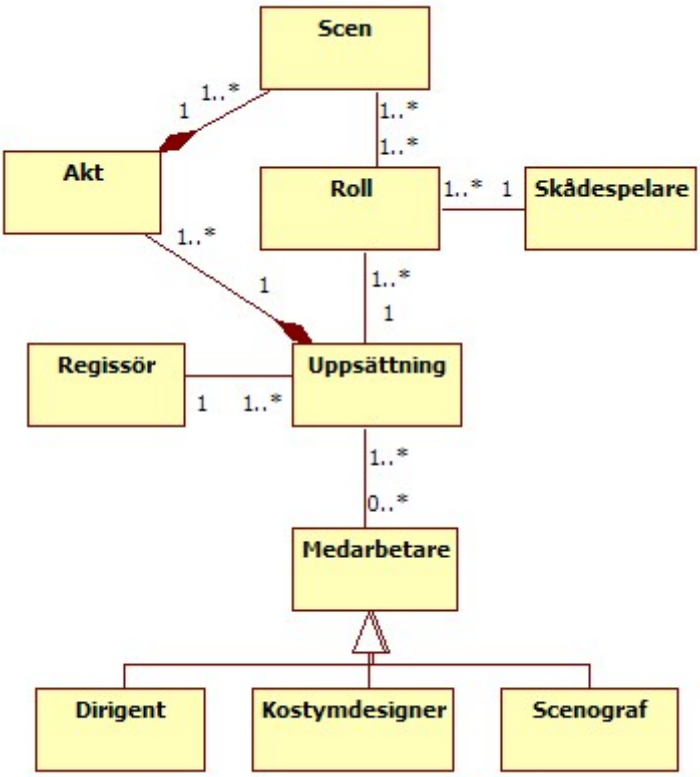
Det är inte nödvändigt att visa attribut eller operationer i klassdiagrammet. Det räcker att skriva ut klassnamn och namn på associationer och multiplicitet. Korrekt UML-notation ska användas.

Systembeskrivning
Teatern GåOchSe vill dokumentera sina teateruppsättningar i ett arkiv där man ska kunna gå in och söka uppgifter om uppsättningar och vem som medverkat i dessa.

Man vill hålla reda på information för varje uppsättning. En teateruppsättning byggs upp av akter som i sin tur byggs upp av scener. Den minsta teaterpjäs man kan sätta upp består alltså av en akt med en scen. I teaterpjäsen finns roller som spelas av skådespelare. En skådespelare kan spela flera roller i samma uppsättning. Därför vill man även hålla reda på vilka roller som medverkar i en viss scen så man veta vilka roller som skulle kunna spelas av samma skådespelare.

Utöver hur teaterpjäsen är uppbyggd och vem som spelar vilken roll så vill man kunna lagra information om övriga inblandade i en uppsättning av en teaterpjäs. Det finns alltid en regissör som man vill hålla reda på vem detta varit. Beroende på uppsättningens storlek så kan där sedan finnas flera andra medarbetare. Dessa kan exempelvis vara kostymdesigners, scenografer, dirigenter med flera.

Lösningsförslag



Uppgift 7 7p

Klassdiagrammet nedan beskriver ett system som hanterar information om kunder och övernattningar för ett hotell.

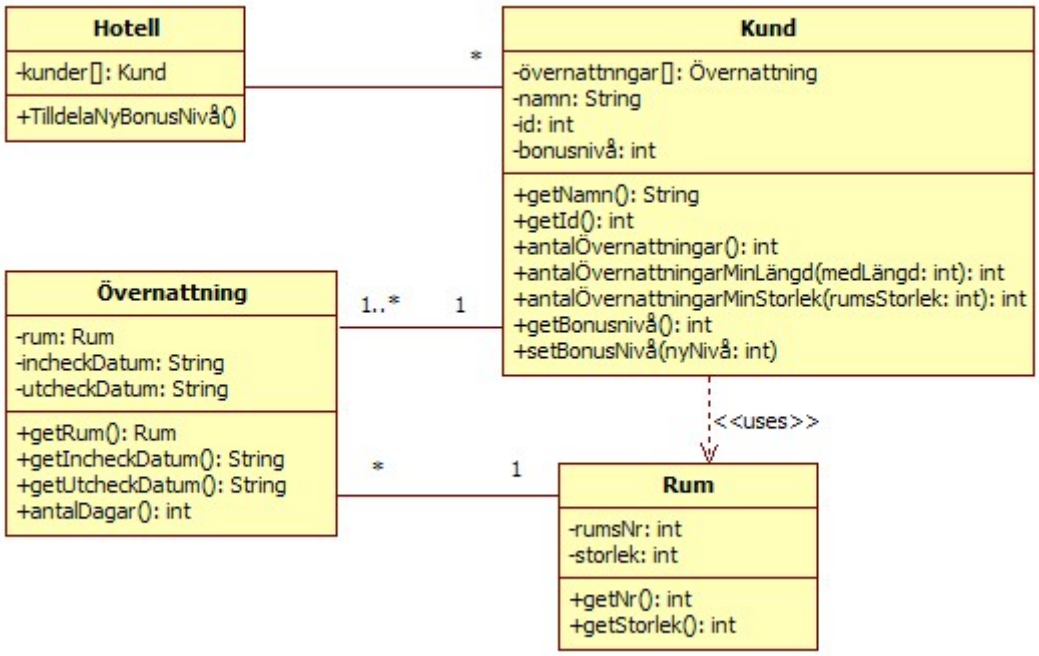
Givet klassdiagrammet nedan skriv ett sekvensdiagram som visar vad som sker när en anställd vill köra den årliga översikten av bonusnivå för kunderna som initieras via metoden TilldelaNyBounusNivå() i klassen Hotell.

Alla kunder som har gjort något av följande

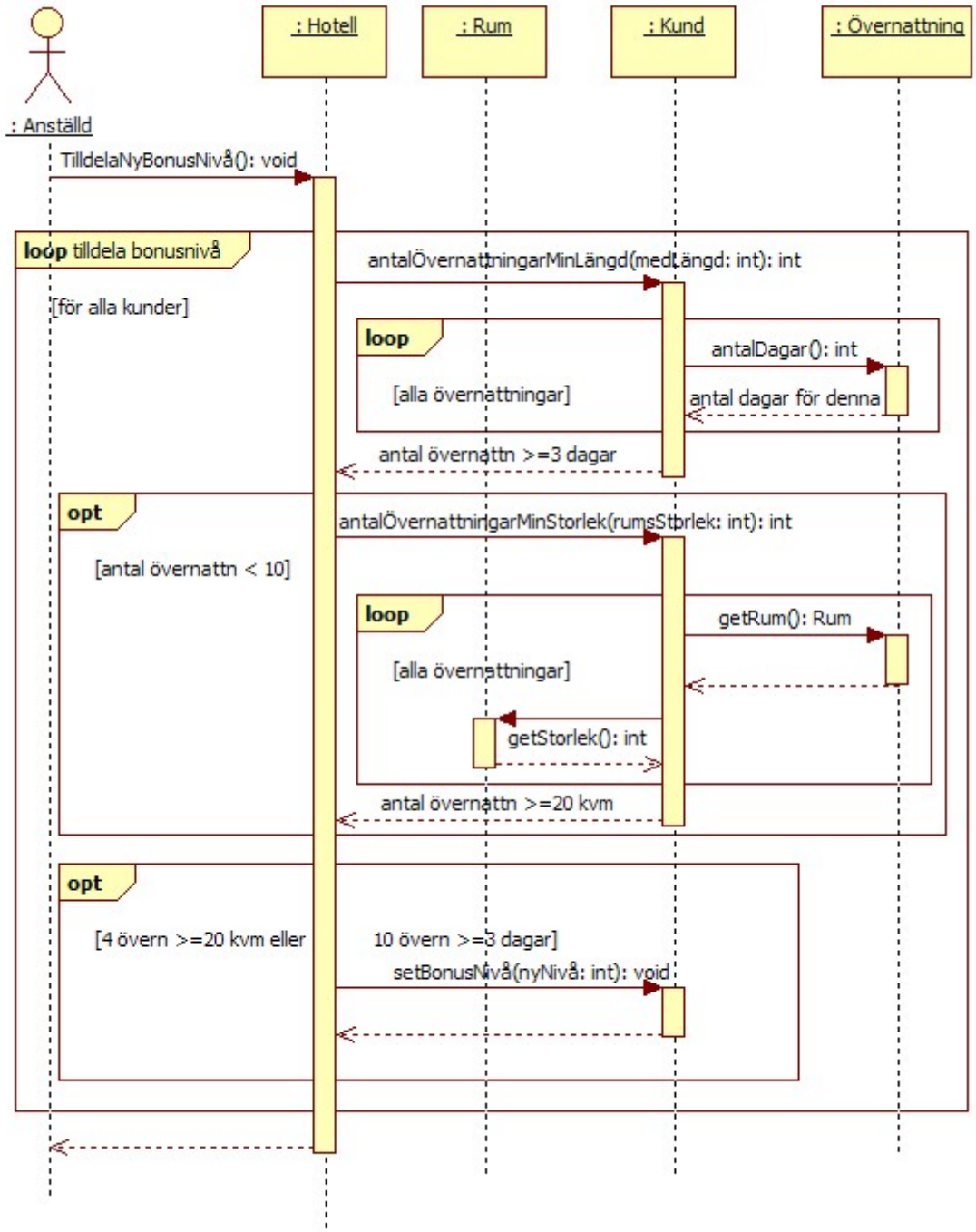
- övernattat minst 10 gånger och dessa övernattningar varat minst 3 dagar
- övernattat minst fyra gånger i ett rum som är minst 20 kvm

ska sättas till bonusnivå 3.

Du får inte lägga till attribut eller operationer till klasserna. Du får däremot skapa en aktör som representerar den anställda personen på hotellet som initierar sekvensen.



Lösningsförslag



Uppgifter för betyg VG

Uppgift 8a

Följande kod är given:

```
class Vehicle
{
    double HorsePower;
    public void changeOil()
    {
        System.out.println("Super class is changing oil");
    }
}
class Car extends Vehicle
{
    public void changeOil()
    {
        System.out.println("Subclass car is changing oil");
    }
}
```

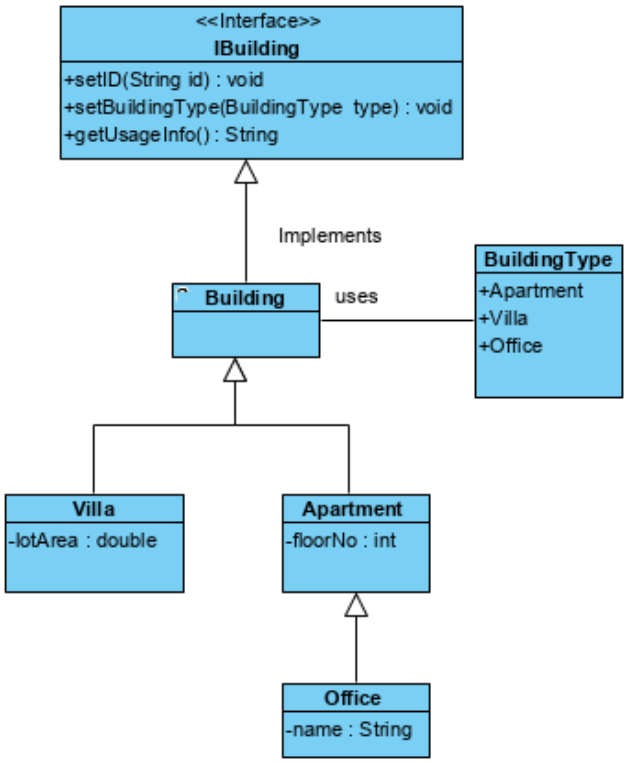
Vilken utskrift ger metoden test i koden nedan när den körs?

```
public void test()
{
    Vehicle vehicle = new Vehicle();
    vehicle.changeOil();
    vehicle = new Car();
    vehicle.changeOil();
}
```

Svar
Super class is changing oil!
Subclass car is changing oil!

Uppgift 8b

I en liten applikation som hanterar några byggnadstyper ingår objekt av typ **Apartment**, **Villa**, och **Office**. För att sätta några regler för subklassernas beteende, skall du skriva ett interface, **IBuilding** som implmenteras av en klass som heter **Building**. Enligt diagrammet ärver **Office** klassen **Apartment** som i sin tur ärver **Building**. **Villa** ärver också **Building**.



Anta att klasserna Villa, Apartment och Office är färdiga men lite andra jobb är kvar till dig att göra. Följ stegen nedan och gör bara det som frågas:

1. Skriv en **public** enum **BuildingType** med de tre byggnadstyperna.
2. Skriv interfacet **IBuilding** med de metoder som framgår av klassdiagrammet.
3. Skriv klassen **Building** som skall implementera **IBuilding**. Implementera alla metoder som **IBuilding** definierar med undantag av att metoden **getUsageInfo** skall vara **abstract**. Anta att denna metod är skriven i klassen **Apartment**, och **Villa**.
4. Utgå ifrån att klasserna Villa, Apartment och Office har setter- och getter- metoder till sina instansvariabler. Skriv metoden **test** (se nedan) färdigt så det går att skapa ett objekt av subklasserna (Office, Apartment och Villa) beroende på typen av byggnad dvs. `buildingType`. Använd de testvärden som anges som kommentarer i koden nedan.

   För att skapa objekten, använd dynamisk bindning. Anropa getUsageInfo och skriv ut returvärdet med hjälp av System.out.println().

```
public void test(BuildingType buildingType)
{
    //Testa följande
```

```
            //ID = ABC100
            //Om buildingType =
            //   Office --> floorNo = 5, name = "Student Union"
            //   Apartment --> floorNo.5
            //   Villa --> lotArea = 194.5  (tomtarea)
        }
```

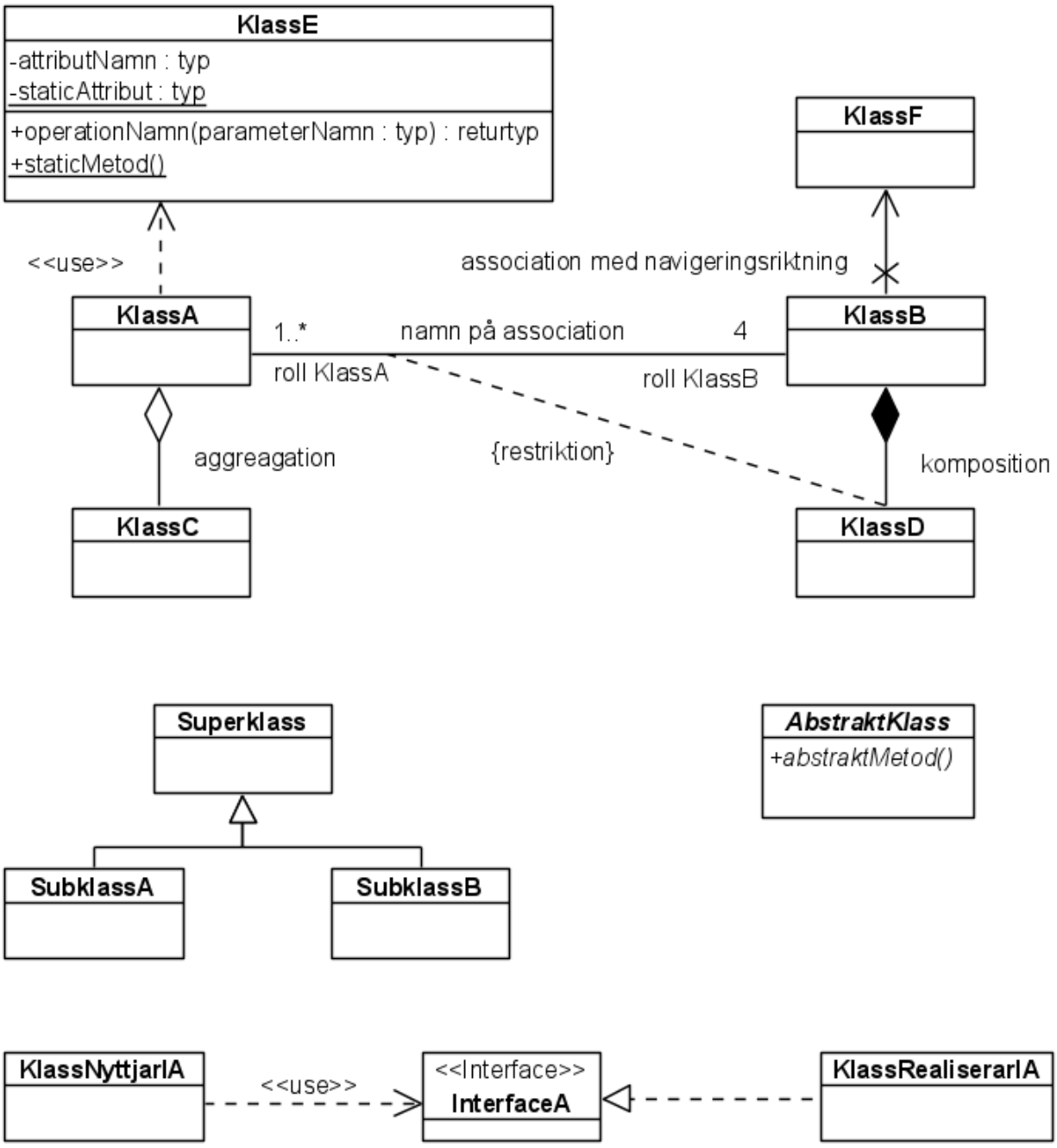
Obs. du ska bara skriva metoden – ingen klass.

svar:

```
public enum BuildingType
{
    Apartment,
    Villa,
    Industry,
    Office
}

public interface IBuilding
{
    void setID(String id);
    void setBuildingType(BuildingType type);
     String getBuildingInfo();
}


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

    public BuildingType getBuildingType()
    {
        return buildingType;
    }

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


    public void setID(String id)
    {
        this.id = id;
    }
    public abstract String getBuildingInfo();
    abstract void DoSomething();
}


public void test(BuildingType buildingType)
{
    //Testa följande
    //ID = ABC100
    //Om buildingType =
    //   Office --> floorNo = 5, name = "Student Union"
    //   Apartment --> floorNo.5
    //   Villa --> lotArea = 194.5  (tomtarea)
```

```
    Building building = null;

    switch (buildingType)
    {
        case Apartment:
            building = new Apartment();
            ( (Apartment) building).setFloorNo(5);
            break;

        case Villa:
            building = new Villa();
            ((Villa)building).setLotArea(194.5);
            break;
        case Office:
            building = new Office();
            ((Office)building).setName("Student Uniion");
            break;
    }
    building.setID("ABC100");
    building.setBuildingType(buildingType);
}
```

Bilaga 1 UML-notation klassdiagram



Bilaga 1 UML-notation sekvensdiagram