UPPSALA
UNIVERSITET

Programmering för språkteknologer 2 – HT 2013

Föreläsning 1

Introduktion och objektorientering

Delvis baserat på slides från Evelina Andersson från 2012

UPPSALA UNIVERSITET

Lärare

- Sara Stymne
- Kontakt
  - sara.stymne@lingfil.uu.se
  - Rum 9-2040
- Om mig
  - Postdoc Uppsala, sedan hösten 2012
  - Doktor, Linköpings universitet, 2012
  - Forskar om maskinöversättning

UPPSALA
UNIVERSITET

Idag

- Introduktion till kursen
  - Kursplan och planering
- Repetition / avstämning
- Objektorientering
  - Klasser
  - Arv
  - Polymorfism

UPPSALA
UNIVERSITET

Kursplan, lärandemål

Efter avslutad kurs skall studenten för att förtjäna betyget Godkänd minst kunna redogöra för följande begrepp och skriva fungerande Javaprogram som exemplifierar och drar nytta av dem:

UPPSALA
UNIVERSITET

Kursplan, lärandemål

- Objektorientering: arv, polymorfism, abstrakta klasser, gränssnitt
- Paket och synlighet
- Hashtabeller och mappningar
- Stackar, köer och länkade listor
- Sökning och sortering
- Matchning med regulära uttryck
- Ändliga automater
- Undantag

UPPSALA
UNIVERSITET

Kursinnehåll

- Två teoretiska huvuddelar:
  - Objektorientering
  - Datastrukturer och algoritmer
- Som även ska kunna appliceras praktiskt
  - Implementera själva
  - Använda och förstå Javas implementationer

UPPSALA UNIVERSITET

Examination – kursmål

Tentamen: Alla lärandemål – framförallt teori

Laborationer: - framförallt praktik

0. Objektorientering, synlighet
1. Regulära uttryck, automater, hashtabeller
2. Sökning, sortering
3. Stackar, köer, listor, undantag, paket
4. Objektorientering, applicering av övriga lärandemål

UPPSALA UNIVERSITET

Betygskriterier

Betyget G:

Minst G på tentamen samt lab 1-4

Betyget VG:

VG på tentamen, samt G på lab 1-4

UPPSALA
UNIVERSITET

Kursplanering

Följer i princip labbarnas innehåll

- v. 36 – Introduktion, objektorientering
  - Fö 1, Lab 0
- v. 37 – Regulära uttryck, automater, hashtabeller, läsning/skrivning
  - Fö 2, Lab 1
- v. 38-39 – Sökning, sortering, generics
  - Fö 3, (4), Lab 2

UPPSALA UNIVERSITET

Kursplanering

- v. (38) 39-41 – Stackar, köer, listor, interface, paket
  - Fö 4-5, Lab 3
- v. 41-44 – Fördjupning, och applicering av kunskaper
  - Fö 6, Lab 4
- v. 45 – Repetition, tentamen
  - Fö 7, tenta

UPPSALA UNIVERSITET

Tentamensdatum

Tentamen 8/11 8-12 (fredag)

Bergsbrunnagatan 15

Glöm inte att anmäla er!

Omtentamen 11/12 8-12 (onsdag)

Fyrislundsgatan 80

UPPSALA UNIVERSITET

Labbar – utförande

- 5 labbar
- Redovisning:
  - lab 1-4 via mail till Sara
  - Lab 0, ingen redovisning
- Labbar görs med fördel i par
  - Ni får själva skapa par
  - Fördel om båda är på ungefär samma nivå
  - Vill man kan man labba individuellt

UPPSALA
UNIVERSITET

Labbar - förberedelser

- På de schemalgda tillfällena finns det handledare
- Utöver detta krävs eget arbete
- För att utnyttja handledning – förbered er!
  - Läs alltid instruktionen innan labtillfället
  - Läs relevant litteratur
  - Börja gärna koda
- Lab 0 i eftermiddag är dock upplagd för att köra igång med direkt

UPPSALA UNIVERSITET

Labbar omfattning

- Labbarna blir efterhand större, och mer omfattande
- Mer tid för större labbar, framförallt lab 3 och 4
- Planera för det!

UPPSALA UNIVERSITET

Labbar - deadlines

- Rekommenderade deadlines
  - Lab 1: 17/9
  - Lab 2: 27/9
  - Lab 3: 14/10
  - Lab 4: 29/10
    - I fas med kursen
    - Snabb rättning garanteras
- Slutdeadline: 8/11
- Uppsamling: 6/12

UPPSALA UNIVERSITET

Kursarbete

- 7.5 hp motsvarar cirka 200 timmars arbete
- Schemalagt – 32 timmar
  - 7*2h fö, 7*2h handledd lab, 1*4h tentamen
- Eget arbete – 168 timmar
  - Arbete med labbar
  - Läsning + tentaplugg
  - Eget arbete med programmering

UPPSALA UNIVERSITET

Kurshemsida

http://stp.lingfil.uu.se/~sara/kurser/pst2/

- Kursinformation
- Schema
- Labinstruktioner
- Föreläsningsmaterial (efterhand)

UPPSALA
UNIVERSITET

Kurslitteratur

- Rekommenderad litteratur
  - Eck, David J. Introduction to Programming Using Java (finns online)
- Det finns mycket alternativ litteratur, några exempel finns på hemsidan
- Även mycket material online
  - Javas API
  - Annat

UPPSALA UNIVERSITET

Tidigare kursutvärderingar

- 2012 – ingen utvärdering
- 2011
  - Helhetsintryck: 4.50
  - Övervägande positiva kommentarer
  - Lite svårt att kombinera med kursen som gick parallellt
    - I år har vi koordinerat deadlines så att de inte ska krocka
    - Lagt in kontinuerliga deadlines för att uppmuntra programmeringsarbete tidigt i kursen, vilket bör göra det mindre tufft i slutet av kursen

UPPSALA
UNIVERSITET

Kursförändringar

- Ny lärare
- Nya labbar
  - Mer fokus på programmeringsövning
  - Mer fokus på objektorientering
  - Regulära uttryck: mer fokus på hur de används i Java, teori finns i andra kurser
- Förändringar baserat på utvärderingar och erfarenheter från liknande kurser
- Huvudmål: meningsfullt, givande inför kommande kurser, kul!

UPPSALA UNIVERSITET

Feedback och utvärderingar

- Utvärderingar under kursen
  - Mittkursutvärdering
  - Slututvärdering
    - Viktigt med feedback!
- Även viktigt med kontinuerlig feedback
  - Säg till om något inte fungerar!
  - Det går ofta att ändra saker under kursens gång

UPPSALA UNIVERSITET

Repetition

- Kortfattad repetition
- Syfte
  - Stämma av vad ni kan sedan tidigare kurs
    - Både för min och er del
  - Säg till om något är oklart eller om ni inte känner igen något!

UPPSALA
UNIVERSITET

Programskal – enkel klass

```
public class Name {

 public static void main(String args[])
 {
   // Kod här!
 }
}
```

Koden behöver sparas i filen Name.java

UPPSALA
UNIVERSITET

Programskal – enkel klass

```
public class Name {

 public static void printNumber(int i) {
    System.out.println(“talet är: “ + i);
 }

 public static void main(String args[])
 {
    printNumber(347);
 }
}
```

UPPSALA
UNIVERSITET

Kompilera och kör

```
$ javac Name.java
$ java Name
$ talet är: 347
```

UPPSALA
UNIVERSITET

Variabler - lokala

```
public class Name{


 public static void main(String args[])
 {
   int i;      //deklarera
   int j = 5; //deklarera med tilldelning
   i = j;      //tilldelning
   j = 28;     //tilldelning
 }
}
```

UPPSALA
UNIVERSITET

Typer

- Enkla typer
  - int, double, char, boolean, ...
- Typer som är klasser
  - String, File, ArrayList, ...

- Vad är skillnaden?

UPPSALA UNIVERSITET

Klasser

- Standardklasser i Java
  - String, File, ArrayList, ...
- Vanliga metoder för dessa:

```
String s;
if (s.length() > 0) {
  String t = s.toLowerCase();
  String[] words = s.split(”\\w+”);
  // ...
}
```

UPPSALA UNIVERSITET

Fält / arrayer

“Emma”	“Åke”	“Nina”	“Sven”	“Kim”

```
String[] namn = new String[5];
//fast storlek

ArrayList<String> = new ArrayList<String>();
//variabel storlek
(implementeras dock som ett fält med fast storlek, som kan behöva ändra storlek ibland)
```

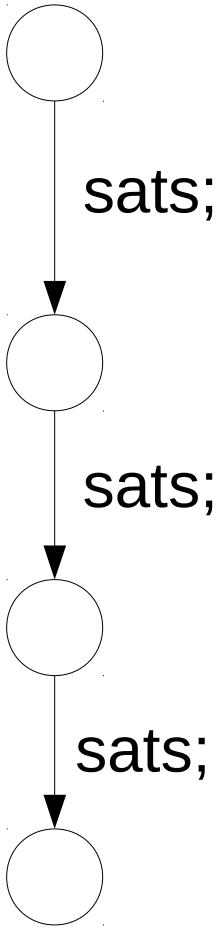
UPPSALA
UNIVERSITET

“Wrapper”-klasser och autoboxing

- “Wrapper”-klasser för att hantera enkla typer
  - Integer – int
  - Double – double
  - ...
- Kan användas när en klass behövs, ex:
  - `ArrayList<Integer> tal`
- Autoboxing / unboxing – konvertera automatiskt:
  - `lista.set(0, 28); //motsvarar`
  - `lista.set(0, new Integer(28));`

UPPSALA
UNIVERSITET

Normalt programflöde



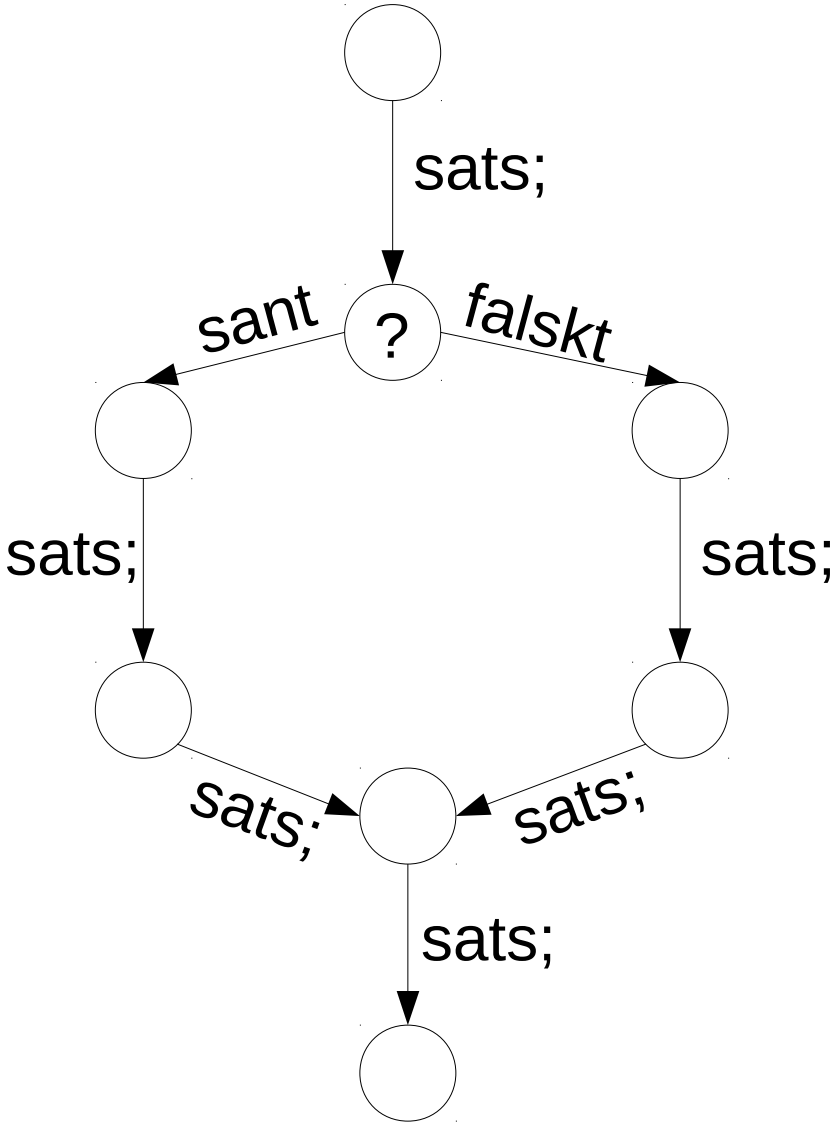
UPPSALA
UNIVERSITET

Normalt programflöde – exempel

```
public class Normal {

   public static void main(String args[])
   {
      int i = 4;
      int j = 5;
      boolean sameNumber = (i == j);

      System.out.println(sameNumber)
   }
}
```

UPPSALA
UNIVERSITET

Villkorsflöde



UPPSALA
UNIVERSITET

Villkorsflöde – exempel 1

```
public class Condition {

 public static void main(String args[])
 {
   int i = 4;
   int j = 5;

   if (i == j) {
     System.out.println("Same number");
   } else {
     System.out.print("Different numbers");
  }
}
```

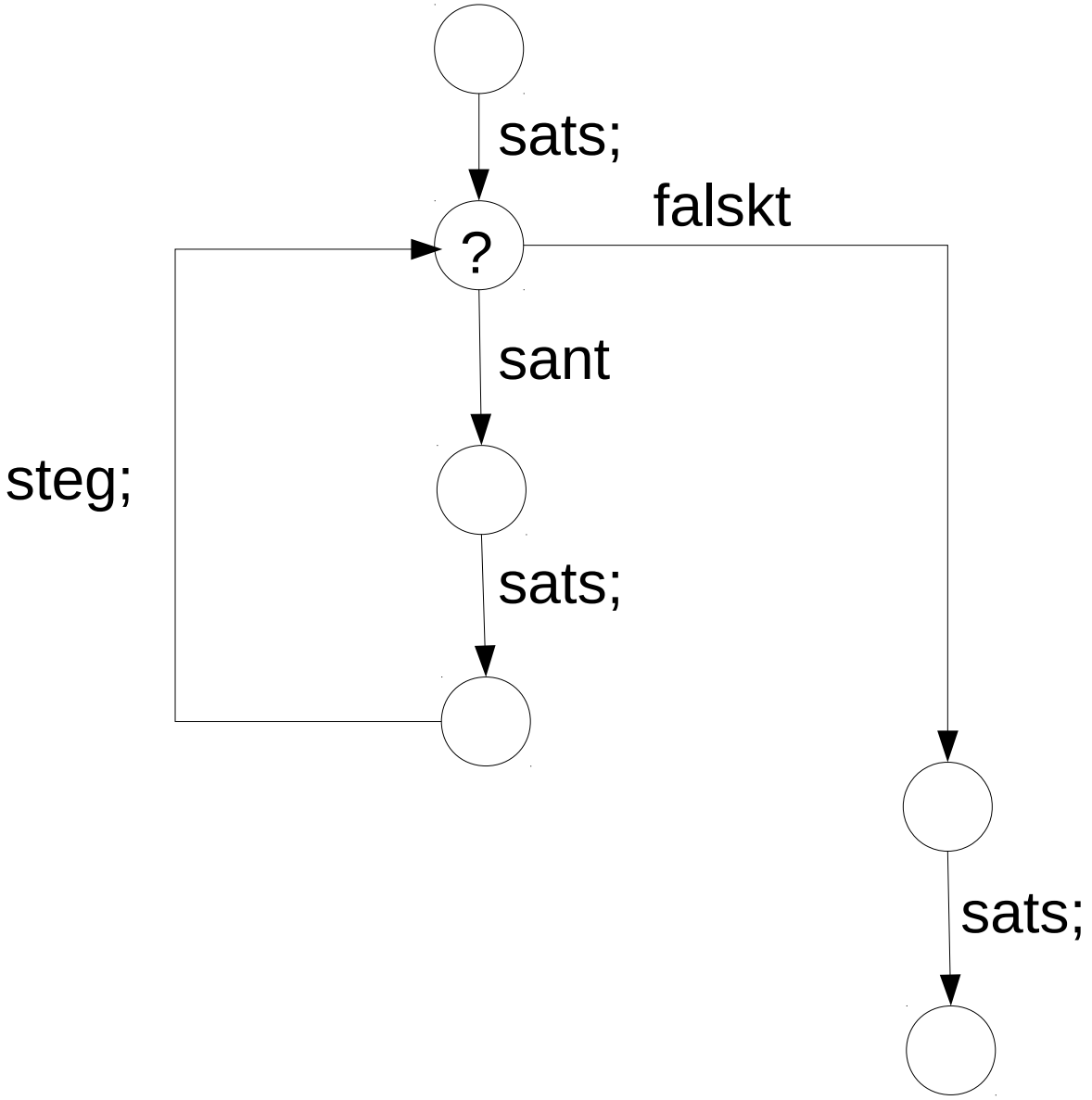
UPPSALA
UNIVERSITET

Villkorsflöde – exempel 2

```
public class Condition {

 public static void main(String args[])
 {
   int i = 4;
   int j = 5;

   if (i == j) {
     System.out.println("Same number");
   } else if (i < j) {
     System.out.println("First number smaller);
   } else {
     System.out.print("First number bigger");
  }
}
```

UPPSALA UNIVERSITET

Upprepningsflöde



UPPSALA
UNIVERSITET

Upprepningsflöde – exempel 1

```
public class While {

   public static void main(String args[])
   {
      int i = 0;

      while(i < 4) {
         System.out.println(i);
         i++;
      }
   }
}
```

UPPSALA
UNIVERSITET

Upprepningsflöde – exempel 2

```
public class For{

   public static void main(String args[])
   {
       for(int i = 0; i < 4; i++)
          System.out.println(i);
   }
}
```

UPPSALA
UNIVERSITET

Upprepningsflöde – array 1

```
public class ForArray1 {

   public static void main(String args[])
   {
       // ...
       for(String s: names)
          System.out.println(s);
   }
}
```

- Vad är names?
- Vad skrivs ut?

UPPSALA
UNIVERSITET

Upprepningsflöde – array 2

```
public class ForArray2 {

  public static void main(String args[])
  {
       ArrayList<String> names = ...

      for(int i = 0; i < s.size(); i++) {
         System.out.println(names.get(i));
  }
}
```

UPPSALA
UNIVERSITET

Upprepningsflöde – array 3

```
public class WhileIterator {

  public static void main(String args[])
  {
      ArrayList<String> names = ...
     Iterator<String> it = names.iterator();
     while(it.hasNext())
     {
        System.out.println(it.next());
     }
  }
}
```

UPPSALA
UNIVERSITET

Kommandoradsargument

```
public class CmdTest {
  Public static void main(String[] args) {
     if (args.length >= 1) {
        System.out.println("Hej " +
                           args[0]);
     } else {
        System.out.println("Vad heter du?");
     }
  }
}

$ java CmdTest Sara
$ Hej Sara
```

UPPSALA
UNIVERSITET

Kodkonventioner

- En beskrivning av hur kod ska se ut
  - Viktigt för att få läsbar kod
  - Konsekvens (speciellt om man är flera)
- Exempel
  - Klassnamn skrivs med stor första bokstav, variabel- och metodnamn med liten första bokstav
  - If-sats ska alltid ha krullparenteser runt sitt kodblock
- För labbarna ska ni följa Oracles kodkonventioner

UPPSALA
UNIVERSITET

Objektorientering

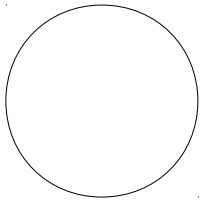
- Klasser
- Instans- och klassvariabler
- Instans- och klassmetoder
- Arv
- Polymorfism
- Synlighet och inkapsling
- UML-diagram

UPPSALA
UNIVERSITET

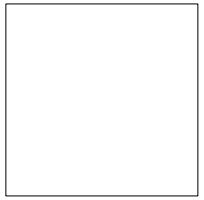
Klasser/Objekt

Klass beskriver objekt

```
class Circle{
....
}
```



```
class Square{
....
}
```

UPPSALA
UNIVERSITET

Exempel - Person

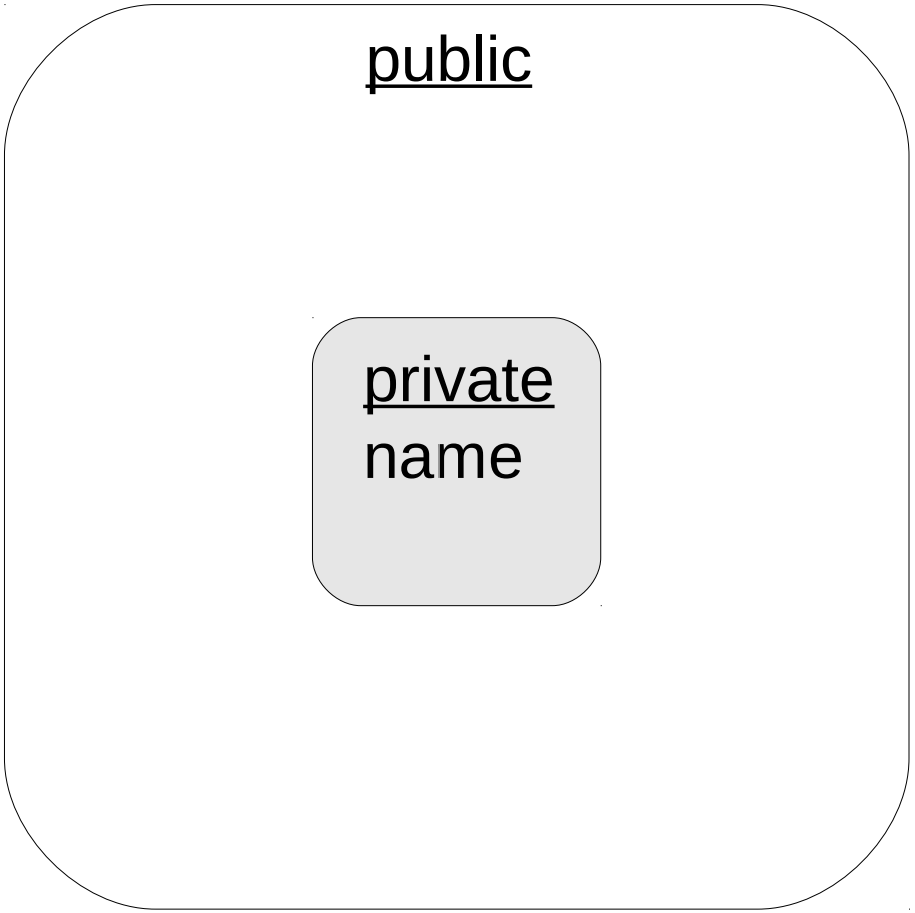
```
public class Person{
    // instance-variable
    private String name = "";

    // constructor
    public Person(String name){
        this.name = name;
    }
}
```

public – synlig för alla

private – endast synlig för klassen

UPPSALA UNIVERSITET

Exempel - Person



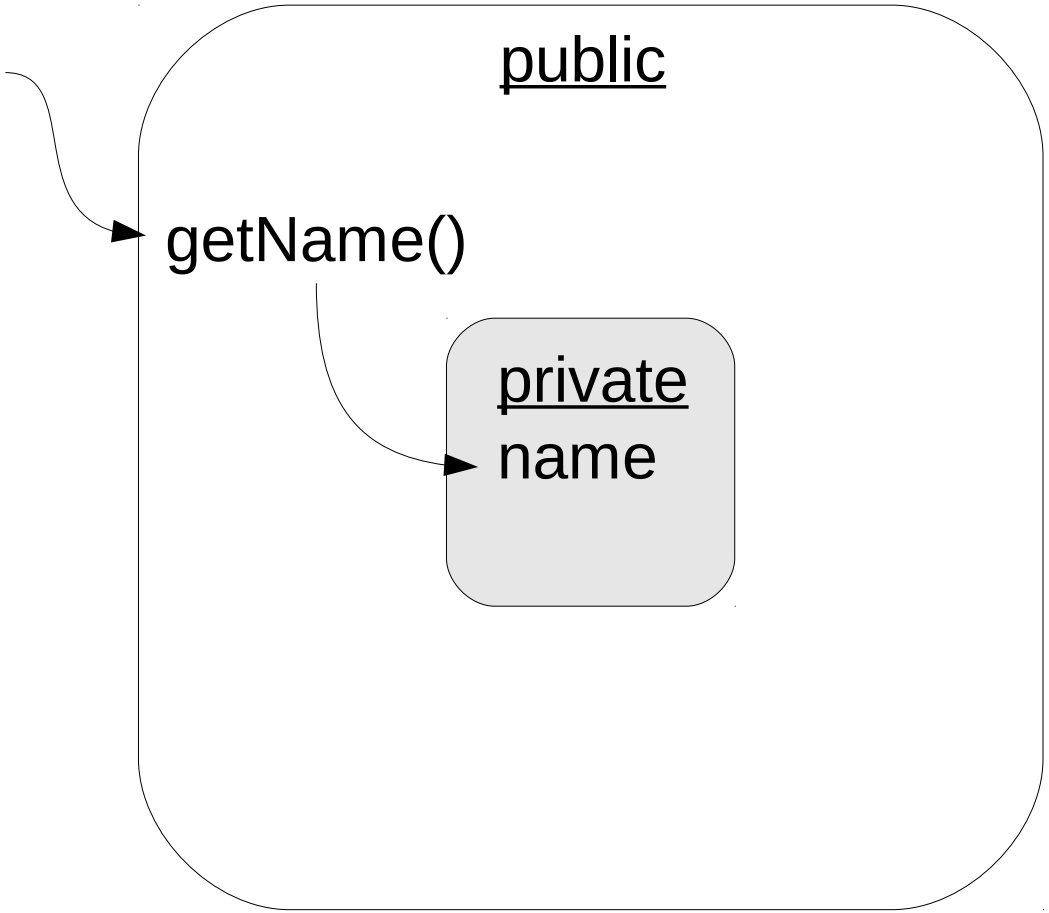
UPPSALA
UNIVERSITET

Exempel - Person

```
public class Person{
    private String name = "";

    // constructor
    public Person(String name){
        this.name = name;
    }

    // method
    public String getName(){
        return name;
    }
}
```

UPPSALA
UNIVERSITET

Exempel - Person



UPPSALA
UNIVERSITET

Exempel - Person

Hur kan vi lägga till ålder?

```
public class Person{
    private String name;

    public Person(String name){
        this.name = name;
    }

    public String getName(){
        return name;
    }
}
```

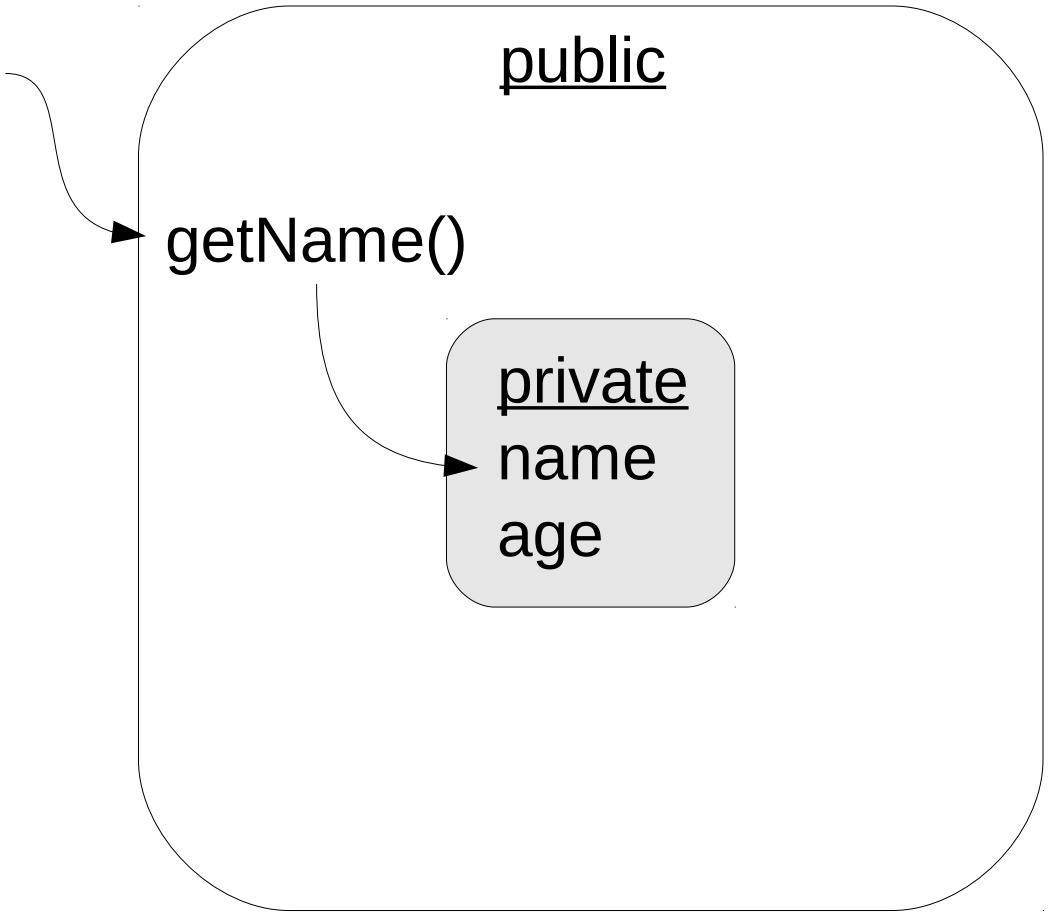
UPPSALA
UNIVERSITET

Exempel - Person

```
public class Person{
    private String name;
    private int age;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public String getName(){
        return name;
    }
}
```

UPPSALA UNIVERSITET

Exempel - Person



UPPSALA
UNIVERSITET

Exempel - Person

Hur kan vi komma åt åldern?

```
public class Person{
    private String name = "";
    private int age = 0;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public String getName(){
        return name;
    }
}
```

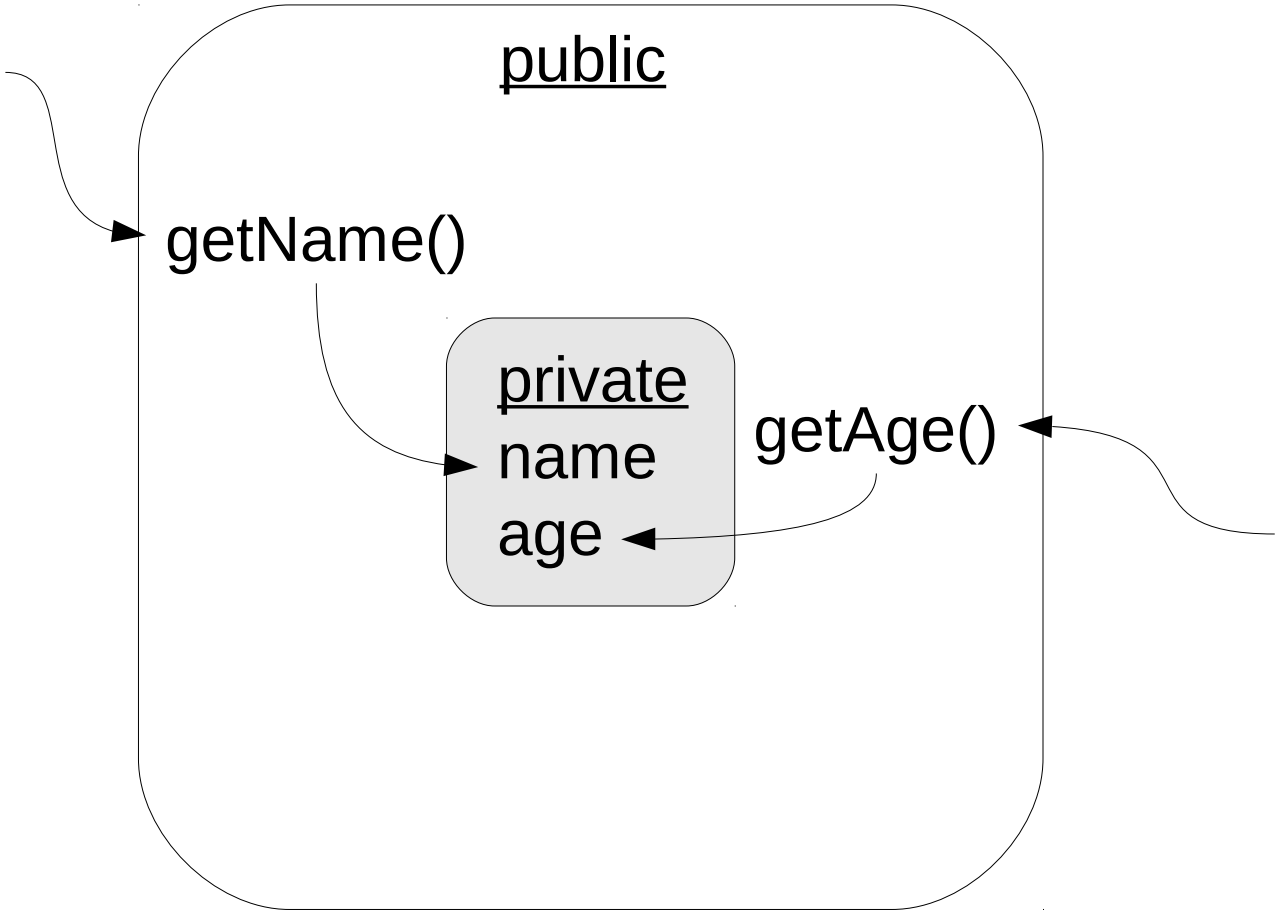
UPPSALA
UNIVERSITET

Exempel - Person

```
public class Person{
    private String name;
    private int age;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    public String getName(){
        return name;
    }

    public int getAge(){
        return age;
    }
}
```

UPPSALA UNIVERSITET

Exempel - Person



UPPSALA
UNIVERSITET

Instansvariabler

Instansvariabel – variabel med ett värde för varje instans av en klass

Exempel i klassen Person:

```
private int name;
private int age;
```

UPPSALA
UNIVERSITET

Inkapsling

- Skydda variabler genom att inte låta andra klasser komma åt dem.
- Deklarera variabler som private
- Använd funktioner för att komma åt och ändra dem:
  - Get-metoder – getName()
  - Set-metoder – setName(String n)
- Håller implementationen gömd
  - lätt att ändra
  - Går att kontrollera värden (exempelvis tillåtna dagar i ett datum)

UPPSALA
UNIVERSITET

Klassvariabler

Klassvariabler - Variabel för hela klassen, finns tillgänglig för varje instans av Klassen

Alla instanser av klassen delar på klassvariablerna.

Exempel:

```
public static int adultAge = 18;
```

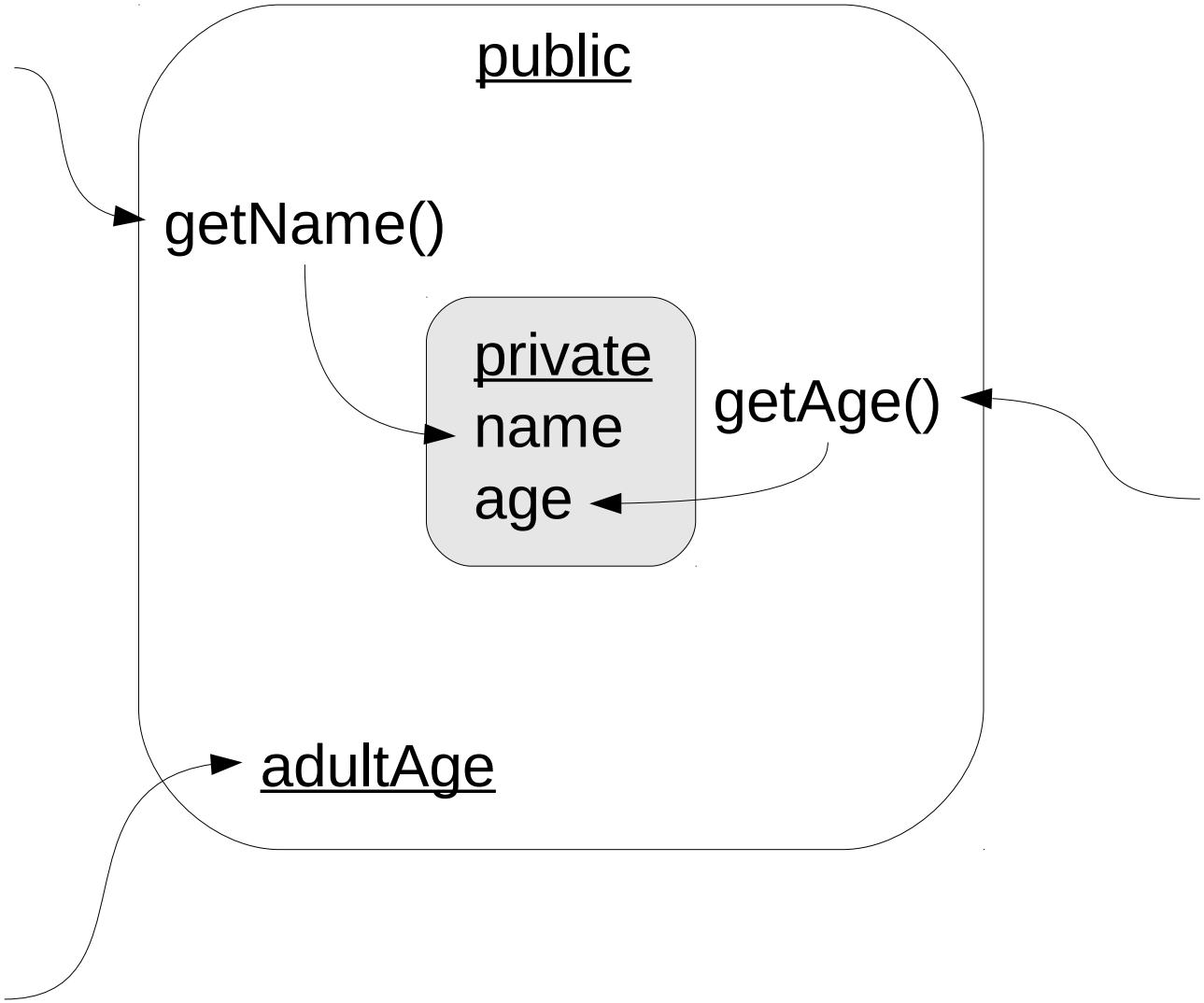
UPPSALA
UNIVERSITET

Klassen Person

```
public class Person{
    public static int adultAge = 18;

    private String name;
    private int age;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    ...
}
```

UPPSALA UNIVERSITET

Exempel - Person



UPPSALA
UNIVERSITET

Klassen Person

Hur kan vi använda adultAge?

```
public class Person{
    public static int adultAge = 18;

    private String name;
    private int age;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }

    ...
}
```

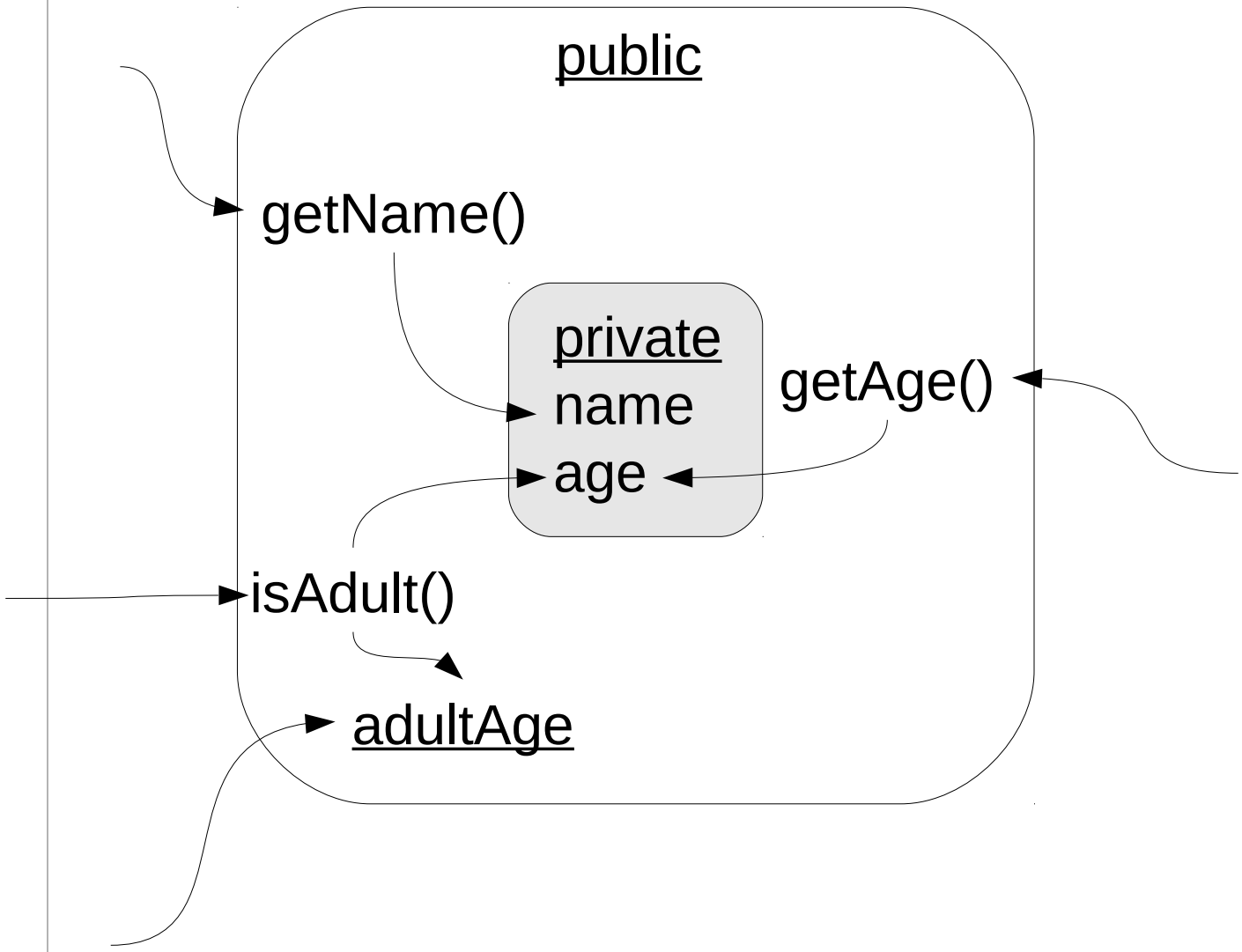
UPPSALA
UNIVERSITET

Klassen Person

```
public class Person{
    public static int adultAge = 18;

    private String name;
    private int age;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }
    ...
    public boolean isAdult() {
       return age == Person.adultAge;
    }
}
```

UPPSALA UNIVERSITET

Exempel - Person



UPPSALA
UNIVERSITET

Klassmetoder

Klassmetod – En metod som endast får använda klassvariabler och inga instansvariabler

Exempel:

```
public static int getAdultAge() {
   return Person.adultAge;
}
```

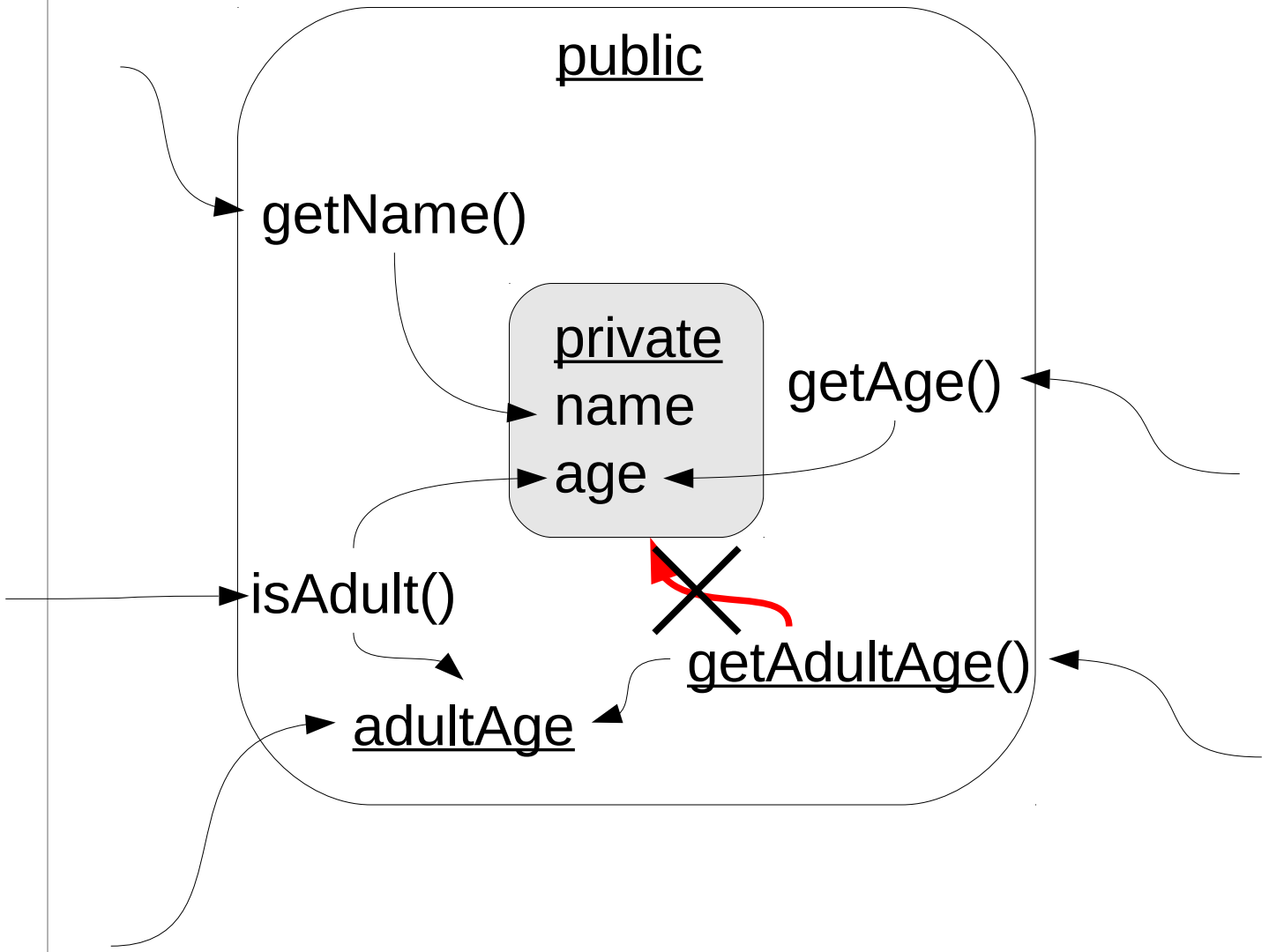
UPPSALA
UNIVERSITET

Klassen Person

```
public class Person{
    public static int adultAge = 18;

    private String name;
    private int age;

    public Person(String name, int age){
        this.name = name;
        this.age = age;
    }
    ...
     public static int getAdultAge() {
       return Person.adultAge;
    }
}
```

UPPSALA UNIVERSITET

Exempel - Person



UPPSALA
UNIVERSITET

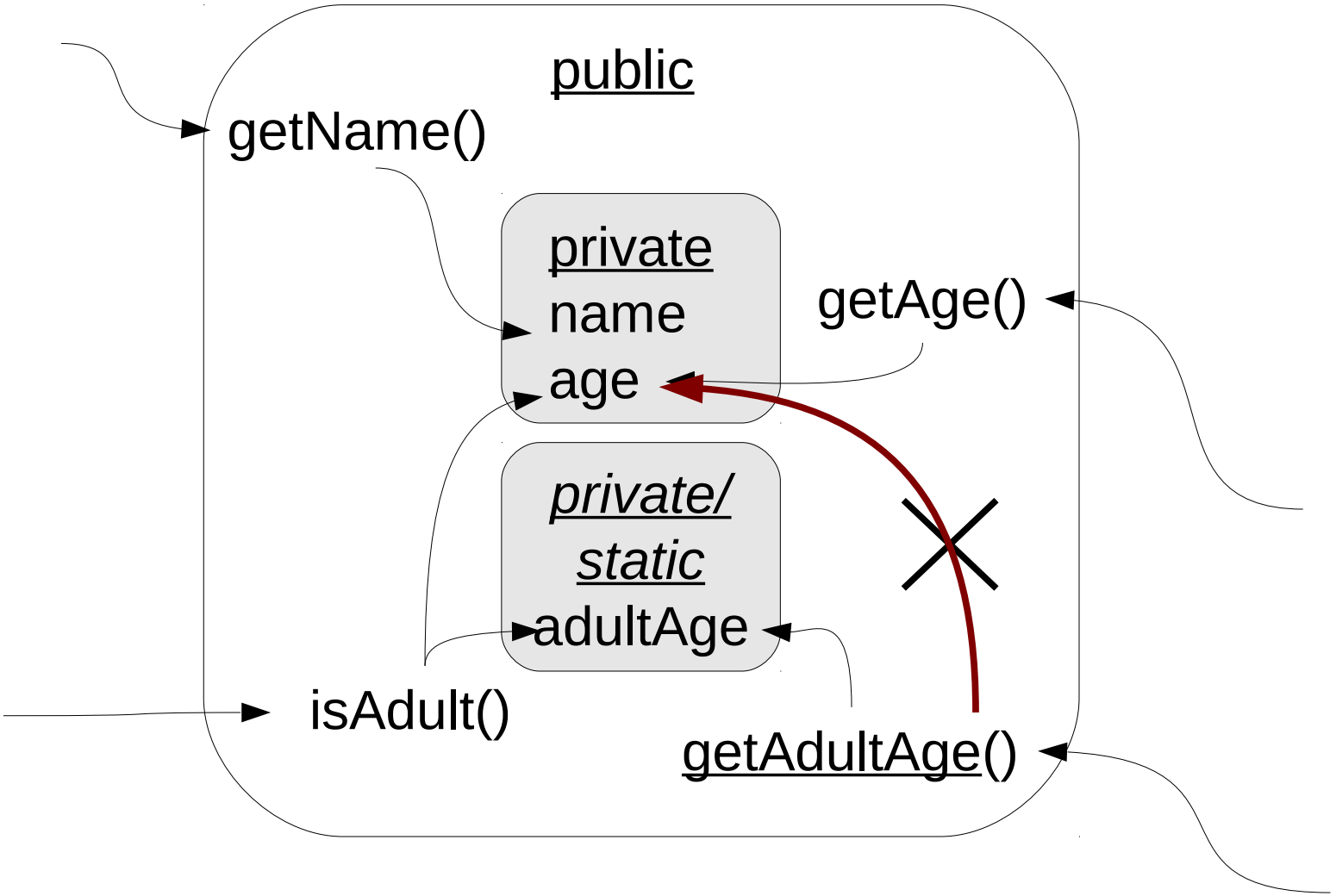
Klassen Person

Borde inte adultAge också vara privat? Jo!

```
public class Person{
    private static int adultAge = 18;

    private String name;
    private int age;

    ...
    public static int getAdultAge() {
       return Person.adultAge;
    }
    public boolean isAdult() {
       return age == Person.adultAge;
    }
}
```

UPPSALA UNIVERSITET

Exempel - Person



UPPSALA
UNIVERSITET

UML

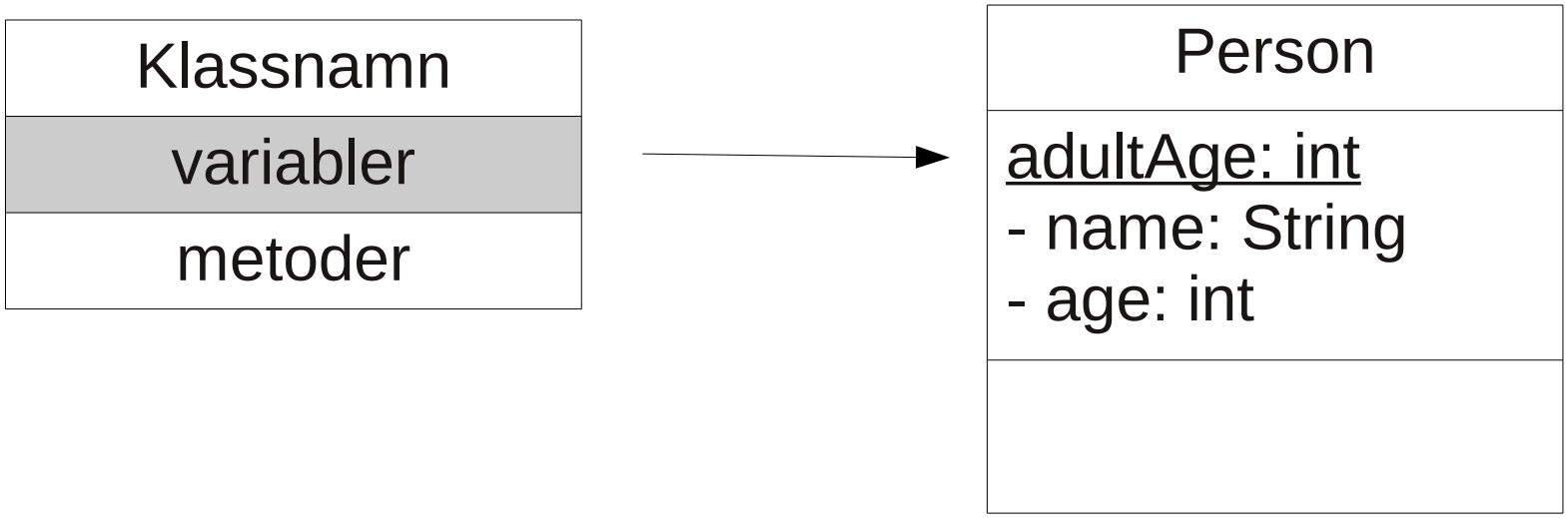
- Unified Modeling Language
- Ett verktyg för att uttrycka idéer
  - Grafisk beskrivning av olika aspekter av objektorienterade system
- Har blivit världsstandard
- Vi kommer att använda klassdiagram
  - Beskriver klasser och relationer mellan klasser

UPPSALA
UNIVERSITET

UML – klassbeskrivning

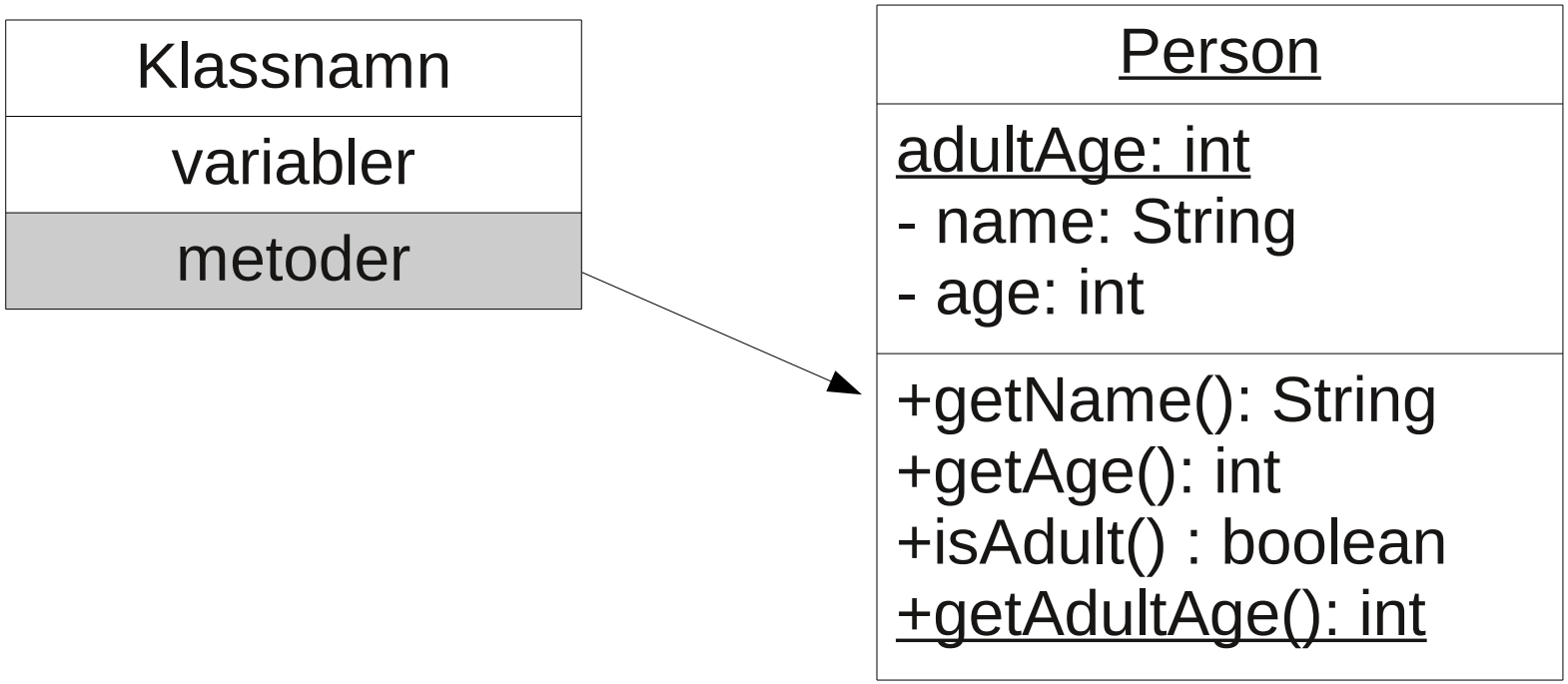
UPPSALA UNIVERSITET

UML – klassbeskrivning



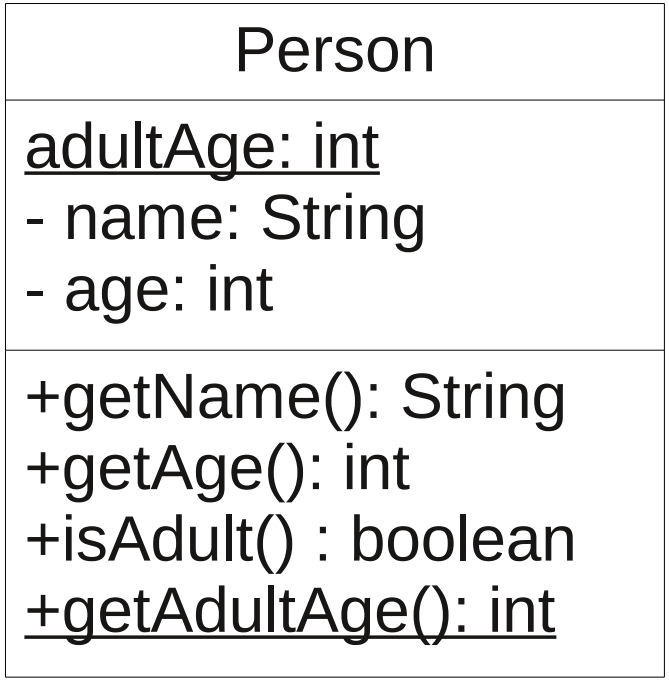
UPPSALA UNIVERSITET

UML – klassbeskrivning



UPPSALA UNIVERSITET

UML – klassbeskrivning



Förklaring

instansvariabel

instansmetod

klassvariabel

klassmetod

- privat

+publik

UPPSALA UNIVERSITET

UML – klassbeskrivning

Person
adultAge: int - name: String - age: int
+getName(): String +getAge(): int +isAdult() : boolean +getAdultAge(): int

Förklaring

instansvariabel

instansmetod

klassvariabel

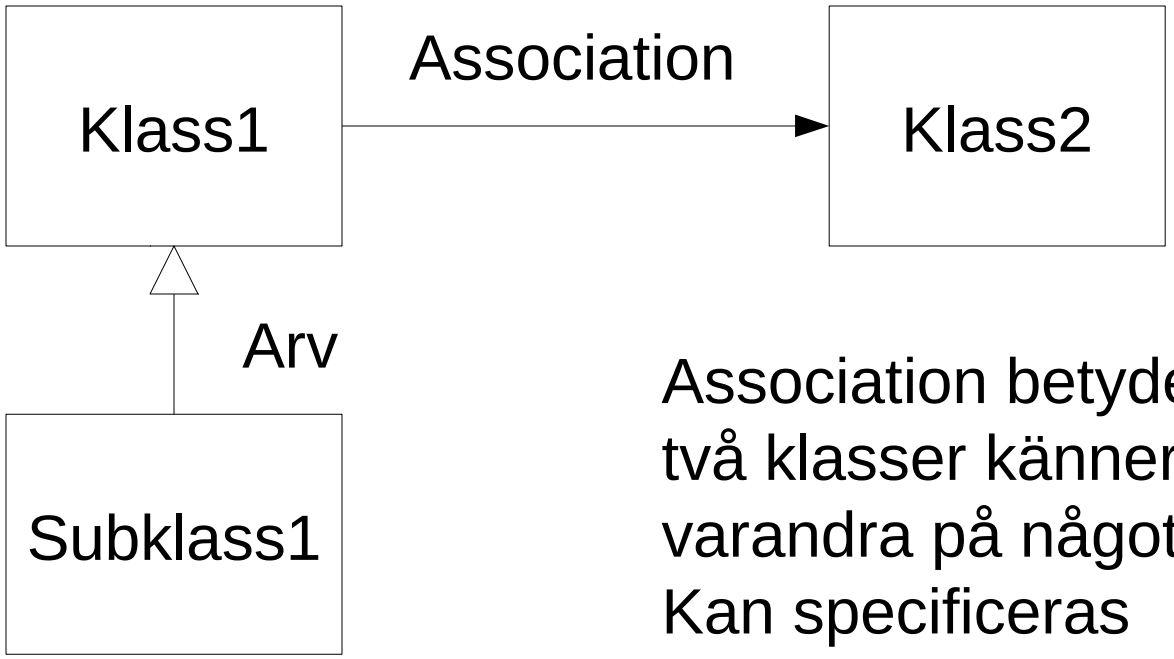
klassmetod

- privat

+publik

UPPSALA UNIVERSITET

UML – relationer mellan klasser



Association betyder att två klasser känner till varandra på något sätt. Kan specificeras ytterligare

UPPSALA
UNIVERSITET

Klassen Car

```
public class Car{
    private Person owner;
    private String regNr;

    public Car(Person myOwner, String myReg){
      owner = myOwner;
      regNr = myReg;
    }
    public String getInfo() {
      return “Car: “ + myReg;
    }
}
```

Car
-owner: Person -regNr: String
+getInfo()

UPPSALA
UNIVERSITET

Klassen Motorcycle

```
public class Motorcycle {
    private Person owner;
    private String regNr;

    public Motorcycle(Person myOwner,
                      String myReg){
      owner = myOwner;
      regNr = myReg;
    }
    public String getInfo() {
      return “Motorcycle: “ + myReg;
    }
}
```

Motorcycle
-owner: Person -regNr: String
+getInfo()

UPPSALA
UNIVERSITET

Klassen Truck

```
public class Truck{
    private Person owner;
    private String regNr;
    private int maxLoad;

    public Truck(person myOwner, String myReg,
                 int myMaxLoad){
      owner = myOwner;
      regNr = myReg;
      maxLoad = myMaxLoad;
    }
    public String getInfo() {
      return “Truck: “ + myReg +
      “, max load: “ + maxLoad;
    }
}
```

Truck
-owner: Person -regNr: String -maxLoad: int
+getInfo()

UPPSALA
UNIVERSITET

Klasserna samlade:

Car
-owner: Person -regNr: String
+getInfo()

Motorcycle
-owner: Person -regNr: String
+getInfo()

Truck
-owner: Person -regNr: String -maxLoad: int
+getInfo()

UPPSALA
UNIVERSITET

Använda klasserna

```
public class ExampleWithoutInheritance{

 public static void main(String args[]) {
   Person owner = new Person("Kalle", 20);
   Car myCar = new Car(owner, "BBC123");
   Motorcycle myMc = new Motorcycle(owner,
                                    "MBC123");
   Truck myTruck = new Truck(owner,
                             "TBC123", 100);

   System.out.println("My name is " +
                      owner.getName() +
                      "\nThese are my vehicles:");
   System.out.println(myCar.getInfo());
   System.out.println(MyMc.getInfo());
   System.out.println(myTruck.getInfo());
  }
}
```

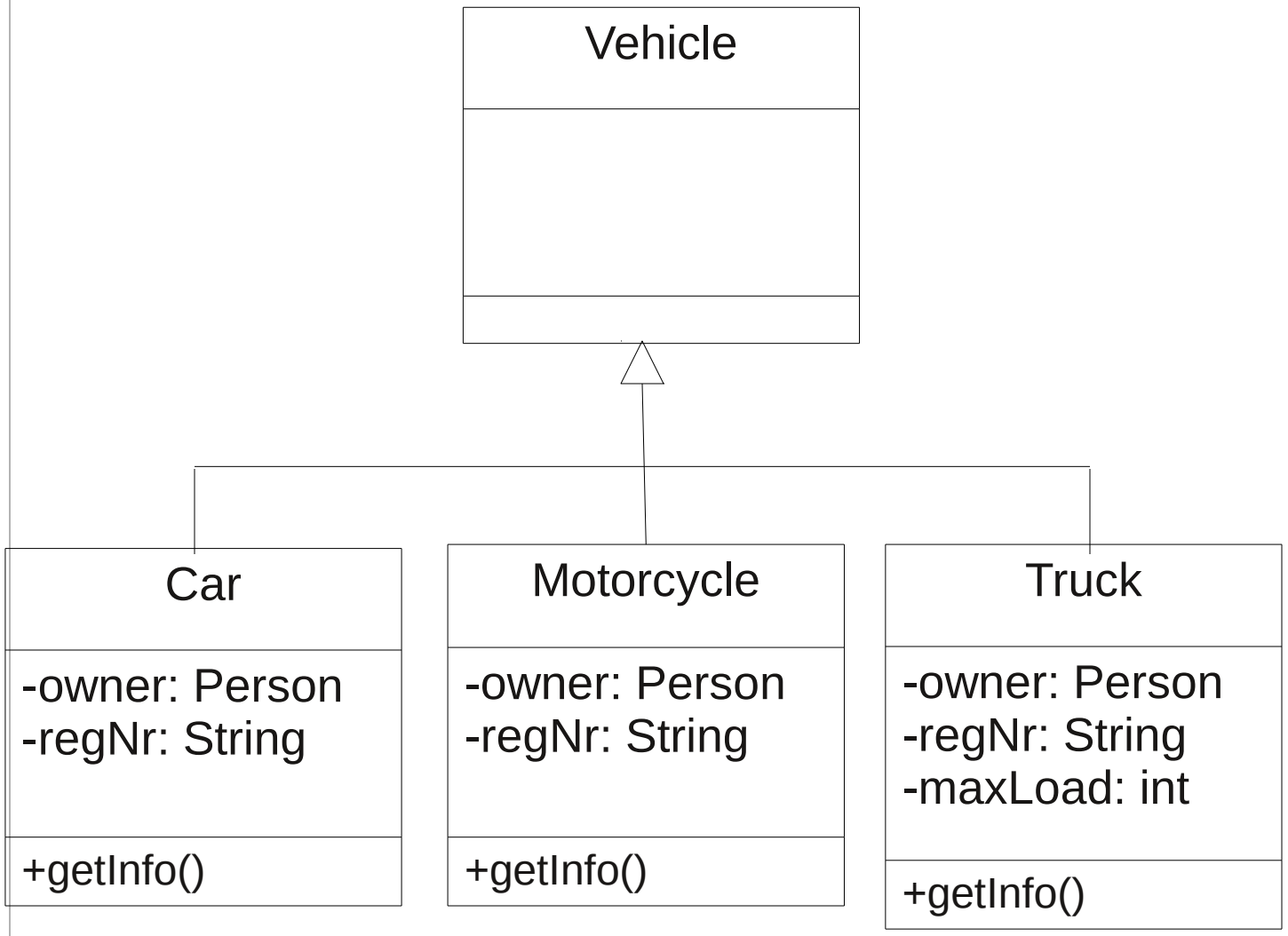
UPPSALA
UNIVERSITET

Provkör!!! - Resultat

```
My name is Kalle
These are my vehicles:
Car: BBC123
Motorcycle: MBC123
Truck: TBC123, max load: 100
```

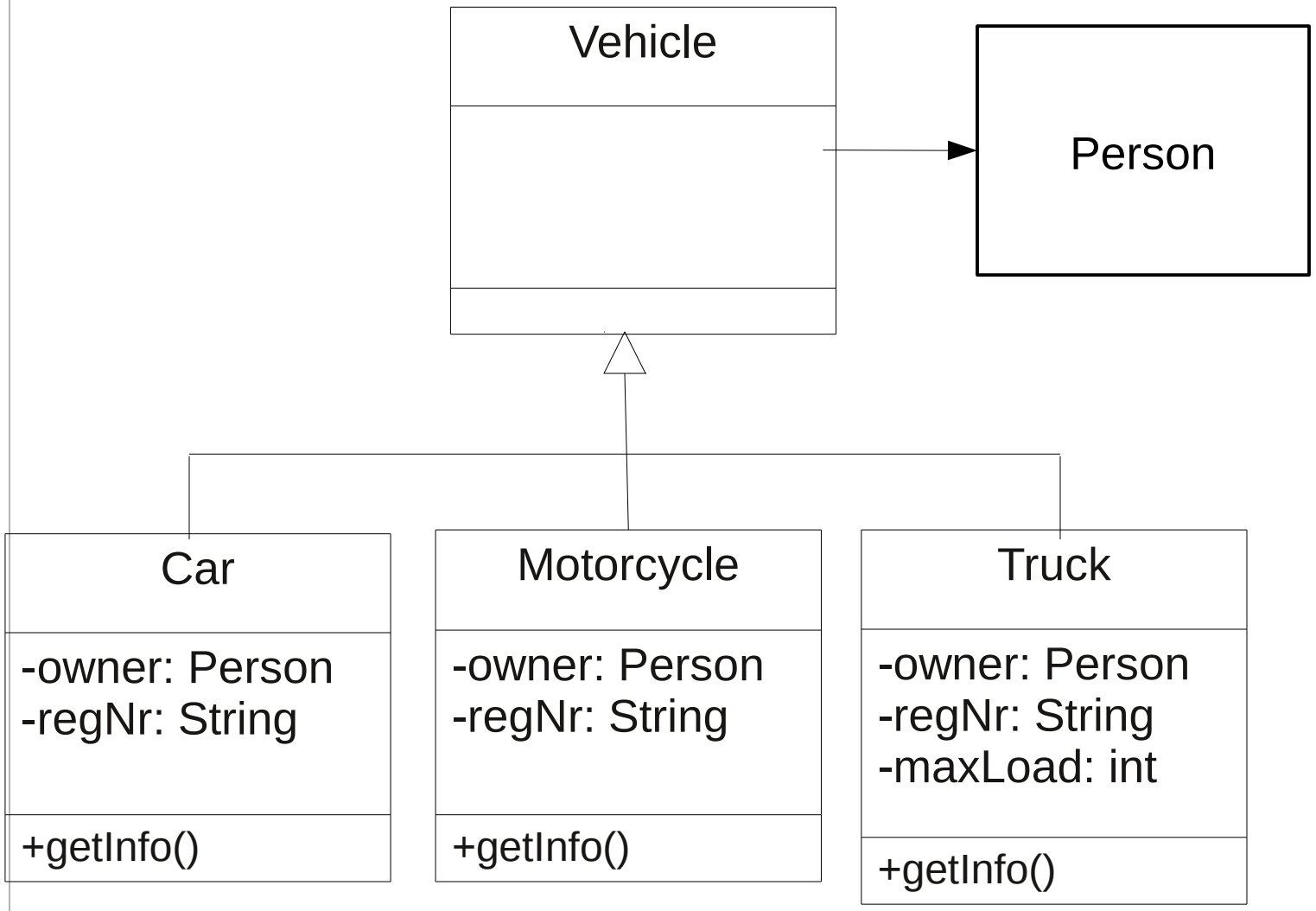
UPPSALA UNIVERSITET

Arv – hur kan vi tala om att klasserna är fordon?



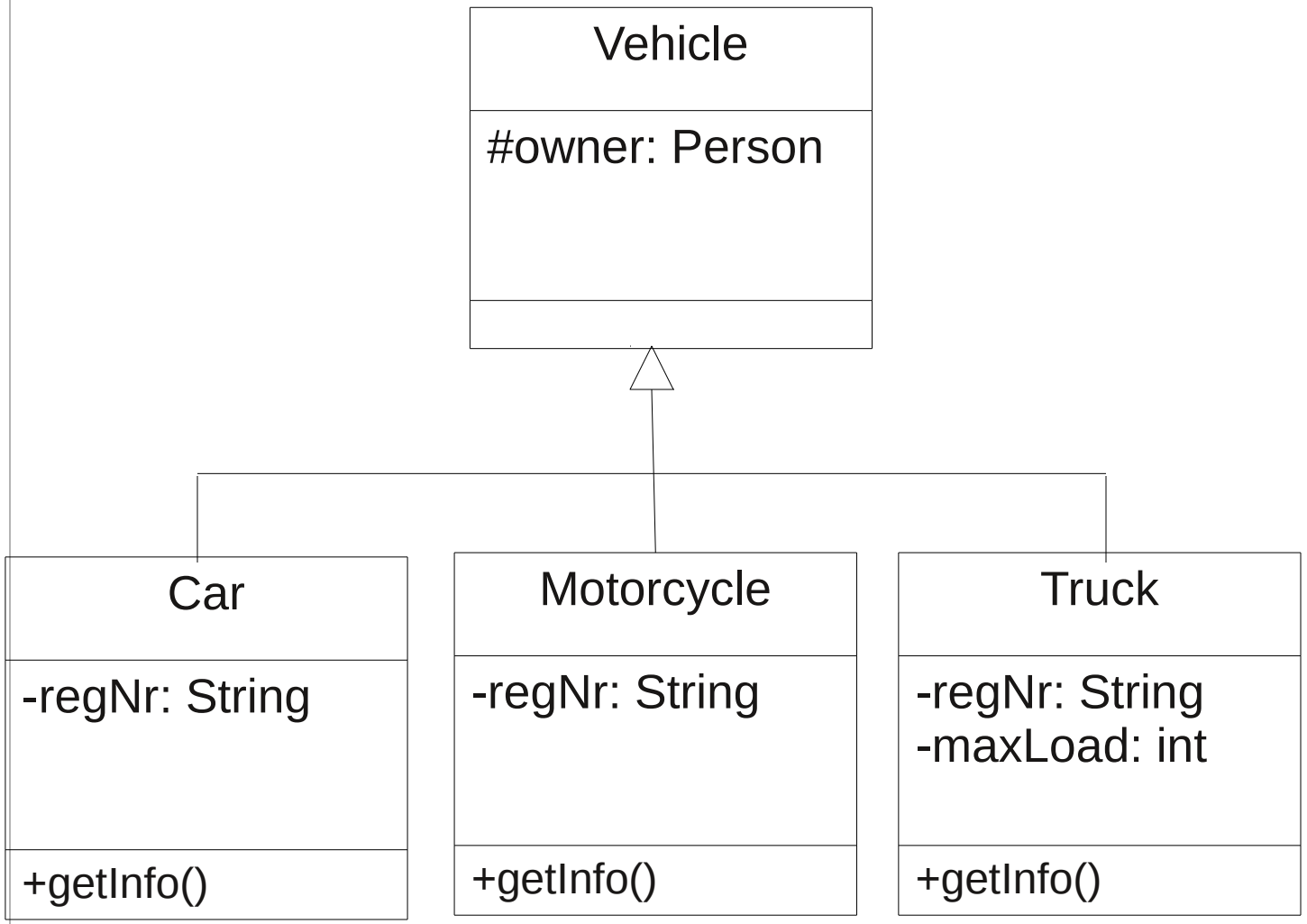
UPPSALA UNIVERSITET

Fordonen känner till klassen Person



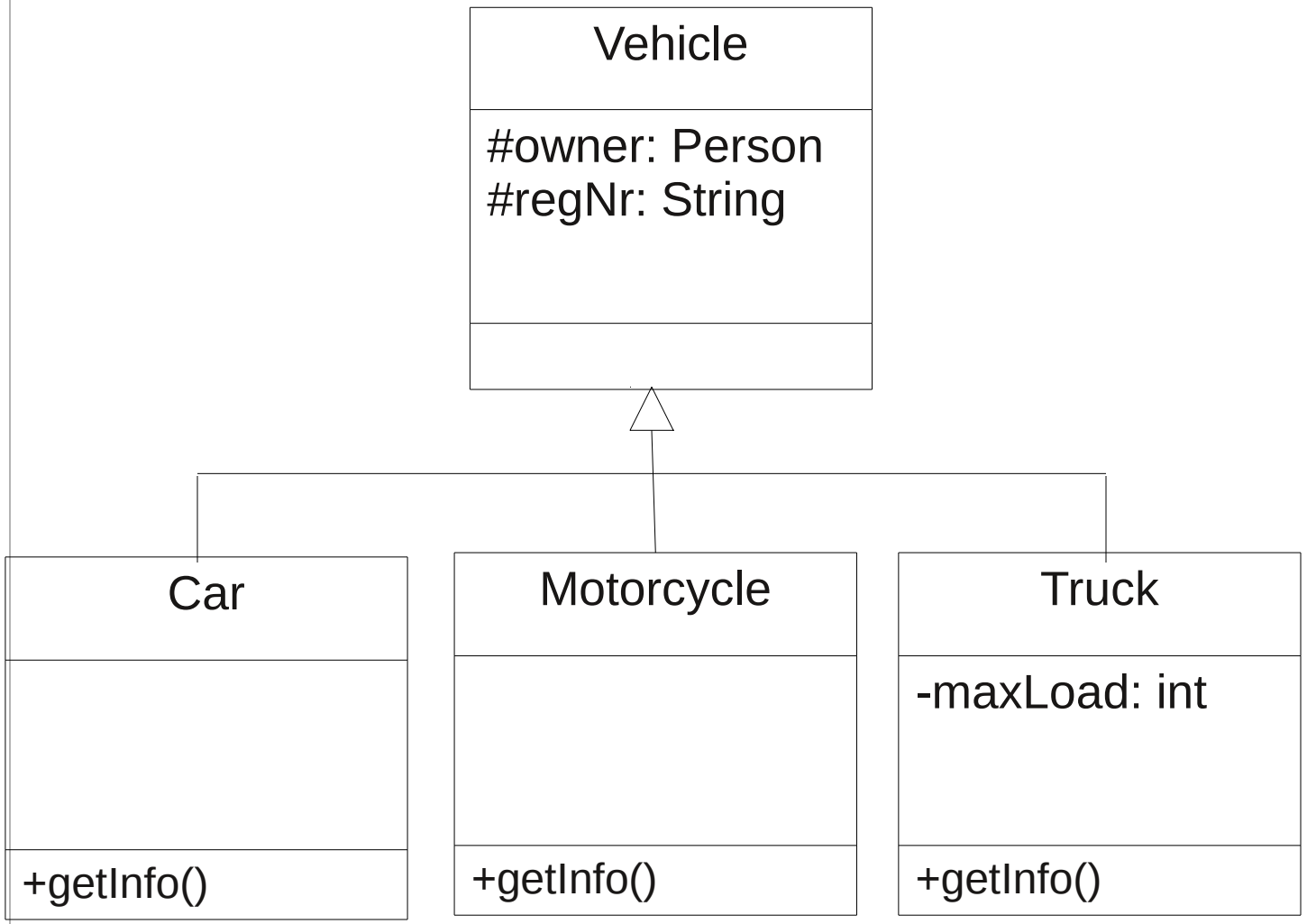
UPPSALA UNIVERSITET

Arv – Vad har alla fordon gemensamt?



UPPSALA UNIVERSITET

Arv – Vad har alla fordon gemensamt?



UPPSALA
UNIVERSITET

Förra koden utan arv

```
public class Car {
    private Person owner;
    private String regNr;
}

public class Motorcycle {
    private Person owner;
    private String regnr;
}

public class Truck {
    private Person owner;
    private String regNr;
    private int maxLoad;
}
```

UPPSALA
UNIVERSITET

Ny kod med arv

```
public class Vehicle {
    protected Person owner;
    protected String regNr;
}

public class Car extends Vehicle {
}

public class Motorcycle extends Vehicle {
}

public class Truck extends Vehicle {
    private int maxLoad;
}
```

Nyckelordet **extends** används när man vill ärva från en annan klass.

UPPSALA
UNIVERSITET

Synlighet

	samma klass	subbklass	alla klasser
public	X	X	X
protected	X	X	
private	X		

UPPSALA
UNIVERSITET

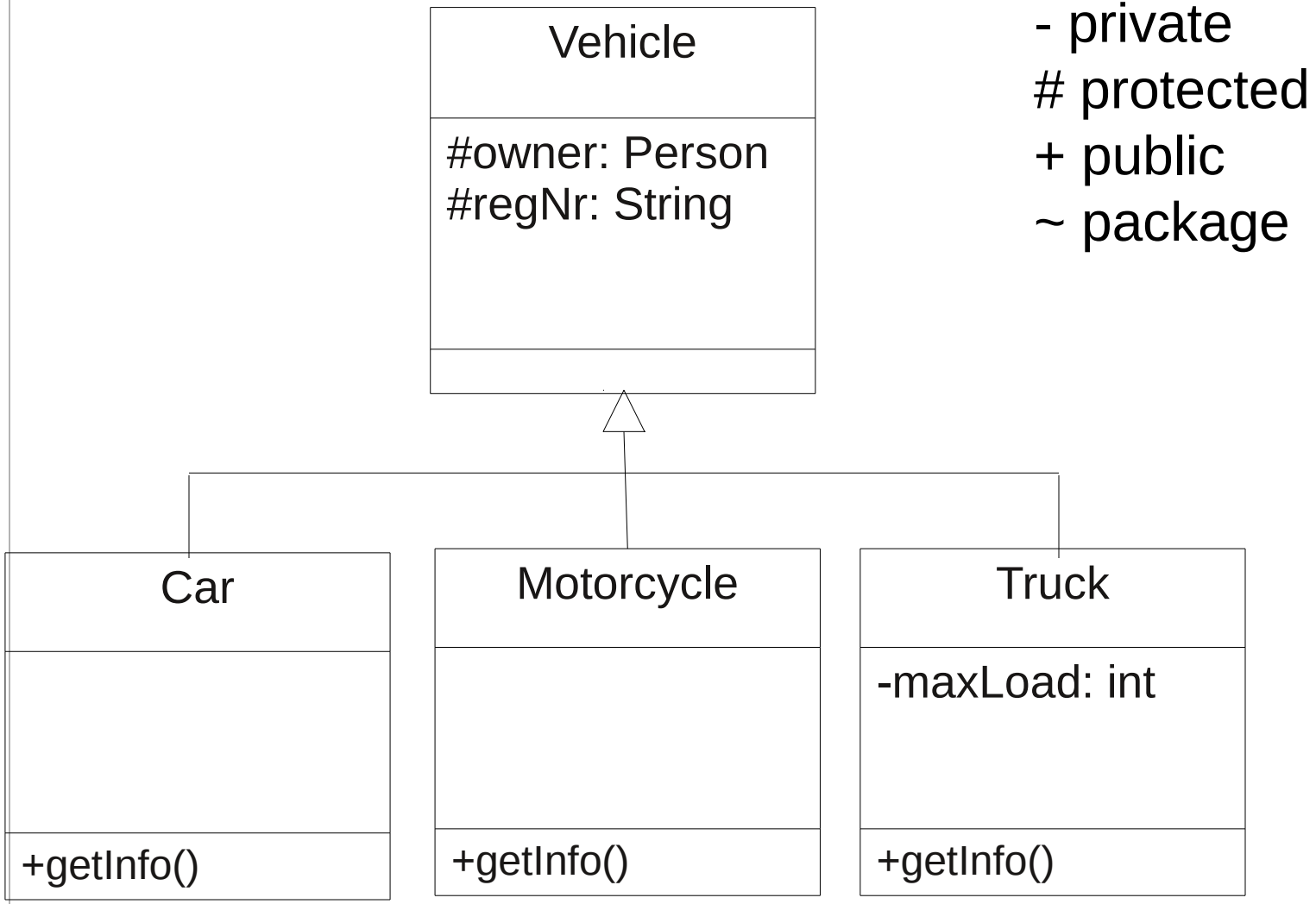
Synlighet – med paket

	samma klass	paket	subklass	alla klasser
public	X	X	X	X
protected	X	X	X	
inget (package)	X	X		
private	X			

Vi kommer att gå igenom paket senare i kursen

UPPSALA UNIVERSITET

Synlighet - UML



- private
protected
+ public
~ package

UPPSALA UNIVERSITET

Ny kod med arv – konstruktorer?

```
public class Vehicle {
    protected Person owner;
    protected String regNr;

    public Vehicle(Person owner,
                   String regNr) {
        this.owner = owner;
        this.name = name;
    }
}

public class Car extends Vehicle {
    public Car(Person owner, String regNr) {
         super(owner, regNr);
    }
}
```

super() används för att anropa konstruktorn för klassen Vehicle

UPPSALA
UNIVERSITET

Ny kod med arv – konstruktorer

```
public class Motorcycle extends Vehicle{
    public Motorcycle(Person owner,
                      String regNr){
        super(owner, regNr);
    }
}

public class Truck extends Vehicle{
    private boolean maxLoad;

    public Truck(Person owner, String regNr,
                 int maxLoad){
        super(owner, regNr);
        this.maxLoad = maxLoad;
    }
}
```

UPPSALA UNIVERSITET

Använda klasserna

```
public class ExampleWithInheritance{

 public static void main(String args[]) {
   Person owner = new Person("Kalle", 20);
   Car myCar = new Car(owner, "BBC123");
   Motorcycle myMc = new Motorcycle(owner,
                                    "MBC123");
   Truck myTruck = new Truck(owner,
                             "TBC123", 100);

   System.out.println("My name is " +
                      owner.getName() +
                      "\nThese are my vehicles:");
   System.out.println(myCar.getInfo());
   System.out.println(MyMc.getInfo());
   System.out.println(myTruck.getInfo());
  }
}
```

UPPSALA
UNIVERSITET

Provkör!!! - Resultat

```
My name is Kalle
These are my vehicles:
Car: BBC123
Motorcycle: MBC123
Truck: TBC123, max load: 100
```

UPPSALA UNIVERSITET

Använda klasserna – alternativ

```
public class ExampleWithInheritance{

 public static void main(String args[]) {
   Person owner = new Person("Kalle", 20);
   Vehicle myCar = new Car(owner, "BBC123");
   Vehicle myMc = new Motorcycle(owner,
                                      "MBC123");
   Vehicle myTruck = new Truck(owner,
                                    "TBC123", 100);

   System.out.println("My name is " +
                        owner.getName() +
                        "\nThese are my vehicles:");
   System.out.println(myCar.getInfo());
   System.out.println(MyMc.getInfo());
   System.out.println(myTruck.getInfo());
  }
}
```

UPPSALA
UNIVERSITET

Provkör!!! - Resultat

```
Problem:

Vehicle does not have any method
called getInfo()
```

Lösning:
lägg till en metod getInfo()
i Vehicle!

Men – vad ska den göra?

UPPSALA UNIVERSITET

Lägg till metod i superklass – lösning 1

```
public class Vehicle {
    protected Person owner;
    protected String regNr;

    public Vehicle(Person owner,
                   String regNr) {
        this.owner = owner;
        this.name = name;
    }

    public String getInfo() {
      return “Fordon: “ + regNr;
    }
}
```

getInfo() i Vehicle kommer nog aldrig att användas

UPPSALA
UNIVERSITET

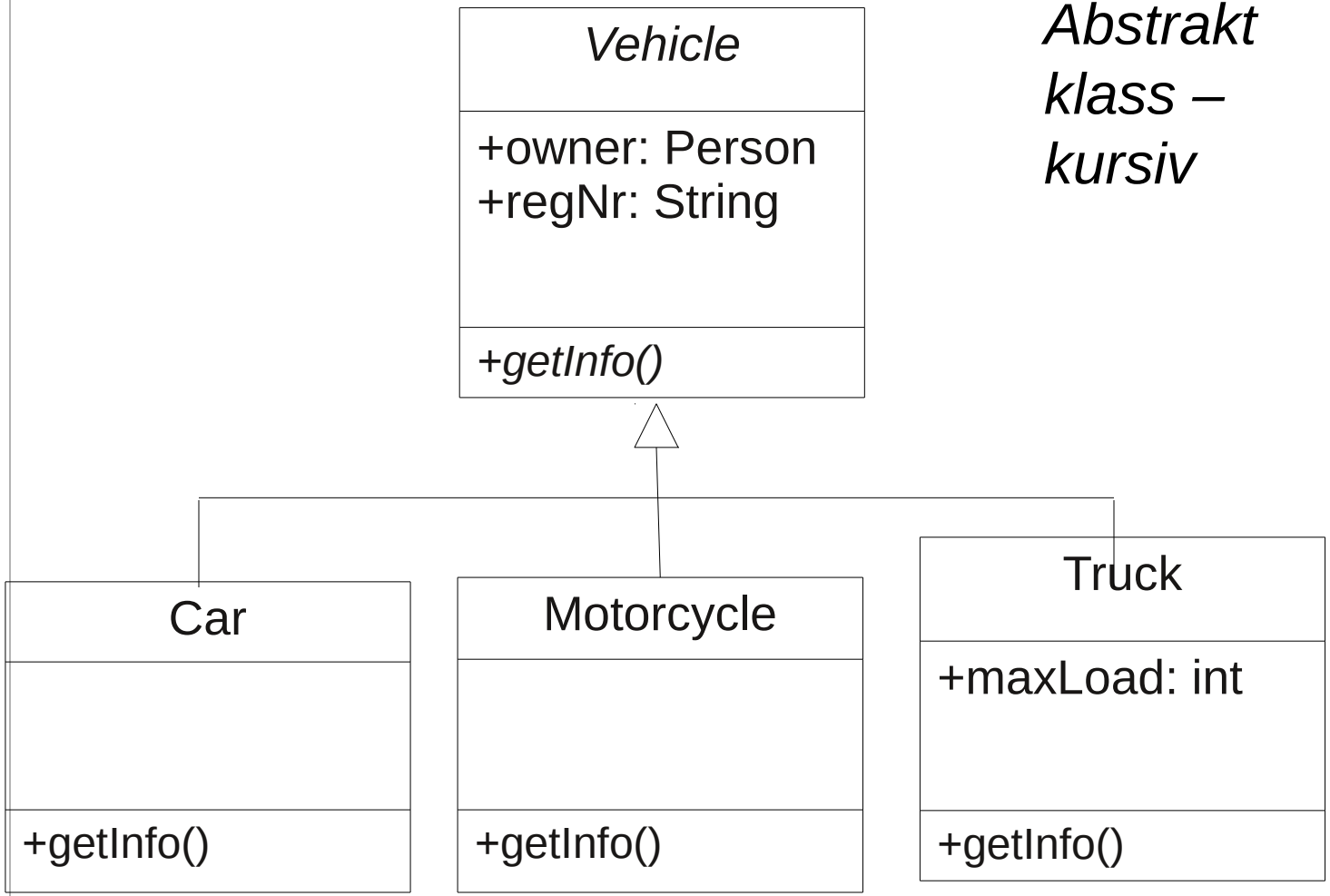
Lägg till metod i superklass – lösning 2

```
public abstract class Vehicle {
    protected Person owner;
    protected String regNr;

    public Vehicle(Person owner,
                   String regNr) {
        this.owner = owner;
        this.name = name;
    }

    public abstract String getInfo();
}
```

Abstrakt klass kan ej finnas som instans
Kan innehålla metoder som är oimplementerade, abstrakta
Ej abstrakta subklasser måste implementera alla metoder!

UPPSALA UNIVERSITET

Abstrakt klass



Abstrakt klass – kursiv

UPPSALA UNIVERSITET

Använda klasserna – alternativ

```
public class ExampleWithInheritance{

 public static void main(String args[]) {
   Person owner = new Person("Kalle", 20);
   ArrayList<Vehicle> vehicles =
                 new ArrayList<Vehicle>();
   vehicles.add(new Car(owner, "BBC123"));
   vehicles.add(new Motorcycle(owner,
                                  "MBC123"));
   vehicles.add(new Truck(owner, "TBC123", 100));

   System.out.println("My name is " +
                        owner.getName() +
                        "\nThese are my vehicles:");
   for (Vehicle v: vehicles) {
      System.out.println(v.getInfo());
   }
  }
}
```

UPPSALA
UNIVERSITET

Provkör!!! - Resultat

```
My name is Kalle
These are my vehicles:
Car: BBC123
Motorcycle: MBC123
Truck: TBC123, max load: 100
```

UPPSALA
UNIVERSITET

Polymorfism

- “mångformighet”
- Metoden i den klass som variabeln har anropas – även om den är deklarerad med en superklass

```
Vehicle myCar = new Car(owner,
                        “BBC123”);

myCar.getInfo();

// Här anropas getInfo() i Car

// ej i Vehicle
```

UPPSALA
UNIVERSITET

Arv - Fakta

Arv

Arv innebär att man skapar nya klasser genom att utgå från redan existerande klasser och utöka dem med ytterligare instansvariabler och metoder.

Exempel:

Klassen Truck är en utökning av klassen Vehicle.

UPPSALA
UNIVERSITET

Arv - Fakta

Superklass - Den klass man ärver från,
till exempel klassen Vehicle.

Subklass – Den klass som ärvt från en superklass,
Till exempel klasserna Car, Motorcycle och Truck

Enkelt arv – När klassen endast ärvt från en superklass

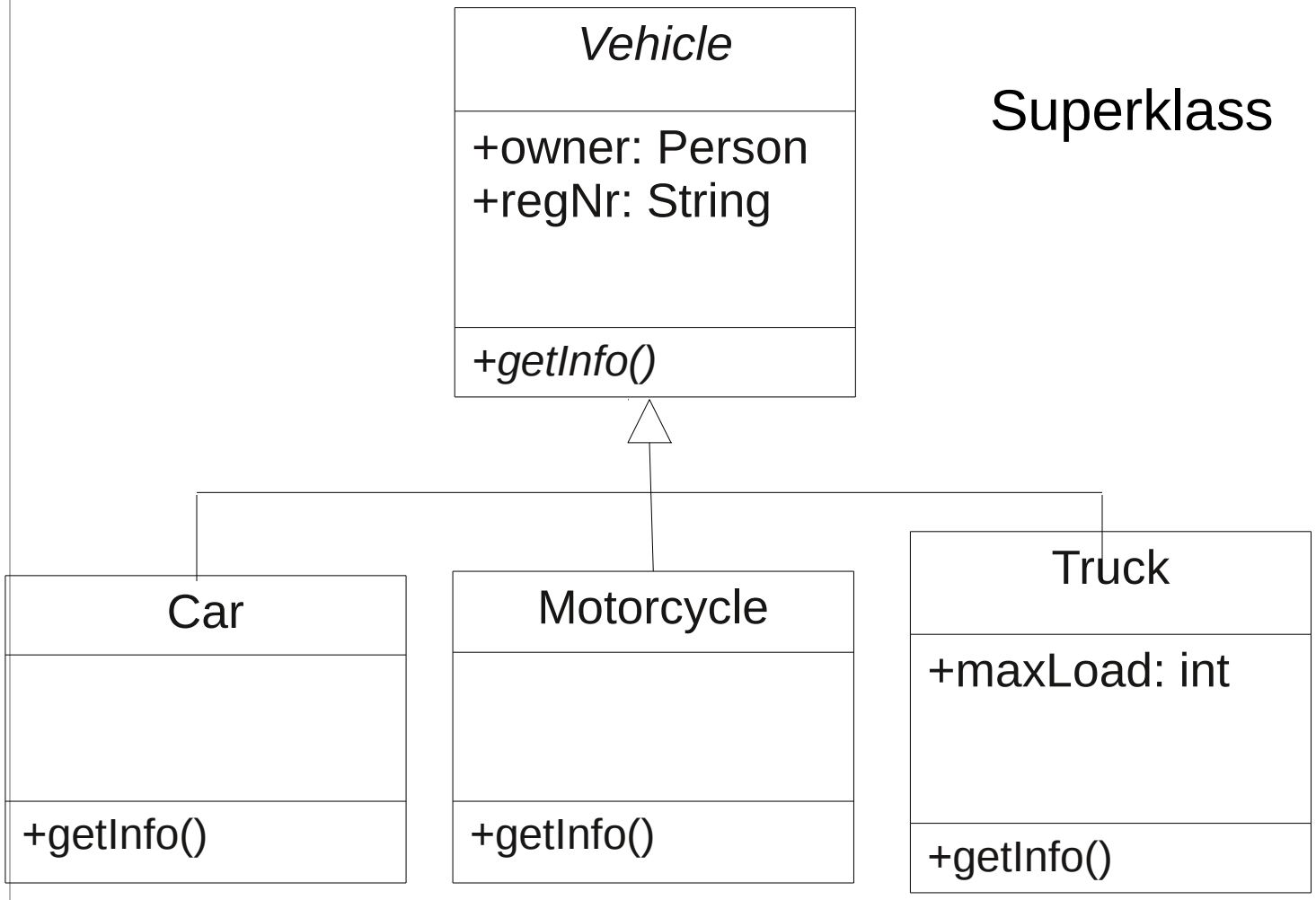
Multipel arv – När klassen ärvt från flera superklasser
(Finns inte i JAVA)

super() - används för att anropa superklassens konstruktor

super.x() – används för att komma åt metod x() i superklassen

UPPSALA UNIVERSITET

Superklass och subklasser



UPPSALA
UNIVERSITET

Arv – Varför?

- Det blir lättare att göra ändringar i rätt klass.

- Programmet blir lättare att underhålla
  och blir mer stabilt

UPPSALA
UNIVERSITET

Hur kan vi byta ägare på ett fordon?

```
public class Vehicle {
  private Person owner;
  private String regNr;
  public void transferOwnerShip(Person newOwner)
   {
     owner = newOwner;
   }
  public String getOwnerName() {
    return owner.getName();
  }
}

public class Car extends Vehicle {
   ...
}

...
```

UPPSALA UNIVERSITET

Main-metod 1

```
public class ExampleWithTransferOwner{
 public static void main(String args[]) {

  Person owner = new Person("Kalle", 20);

  Vehicle myCar = new Car(owner, "BBC123");
  Truck myTruck =
        new Truck(owner, "TBC123", 100);

  System.out.println(myCar.getOwnerName() +
                     " owns the Car.");
  System.out.println(myTruck.getOwnerName() +
                     " owns the Truck. \n");
```

UPPSALA UNIVERSITET

Main-metod 2

```
System.out.println("Stina buys the Truck from "+
                   myTruck.getOwnerName()+
                   ".");

myTruck.transferOwnerShip
              (new Person("Stina",18));

System.out.println("");

System.out.println(myCar.getOwnerName() +
                   " owns the Car.");

System.out.println(myTruck.getOwnerName() +
                    " owns the Truck.");
}
```

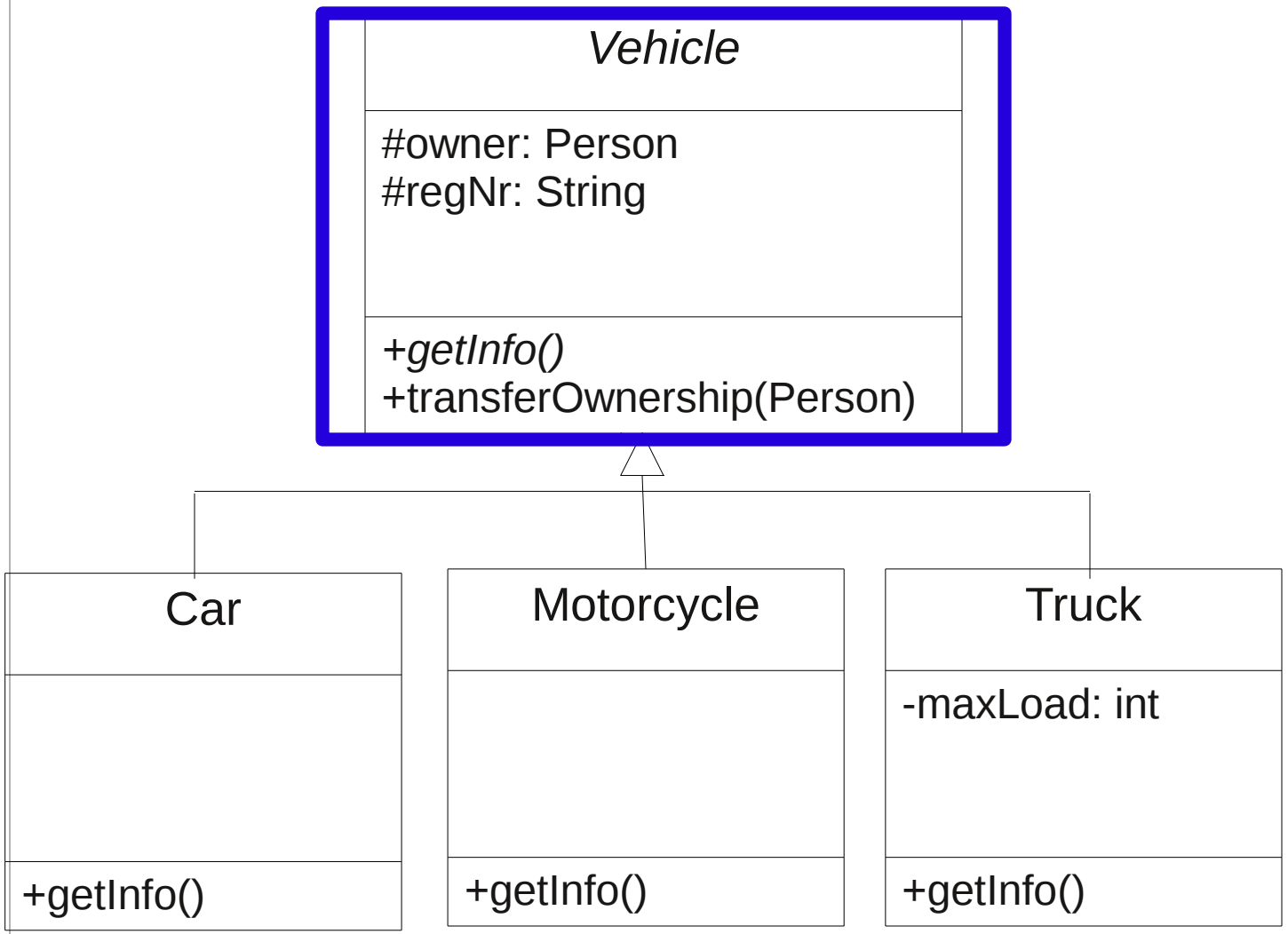
UPPSALA
UNIVERSITET

Provkör!!! - Resultat

```
Kalle owns the Car.
Kalle owns the Truck.

Stina buys the Truck from Kalle.

Kalle owns the Car.
Stina owns the Truck.
```

UPPSALA UNIVERSITET

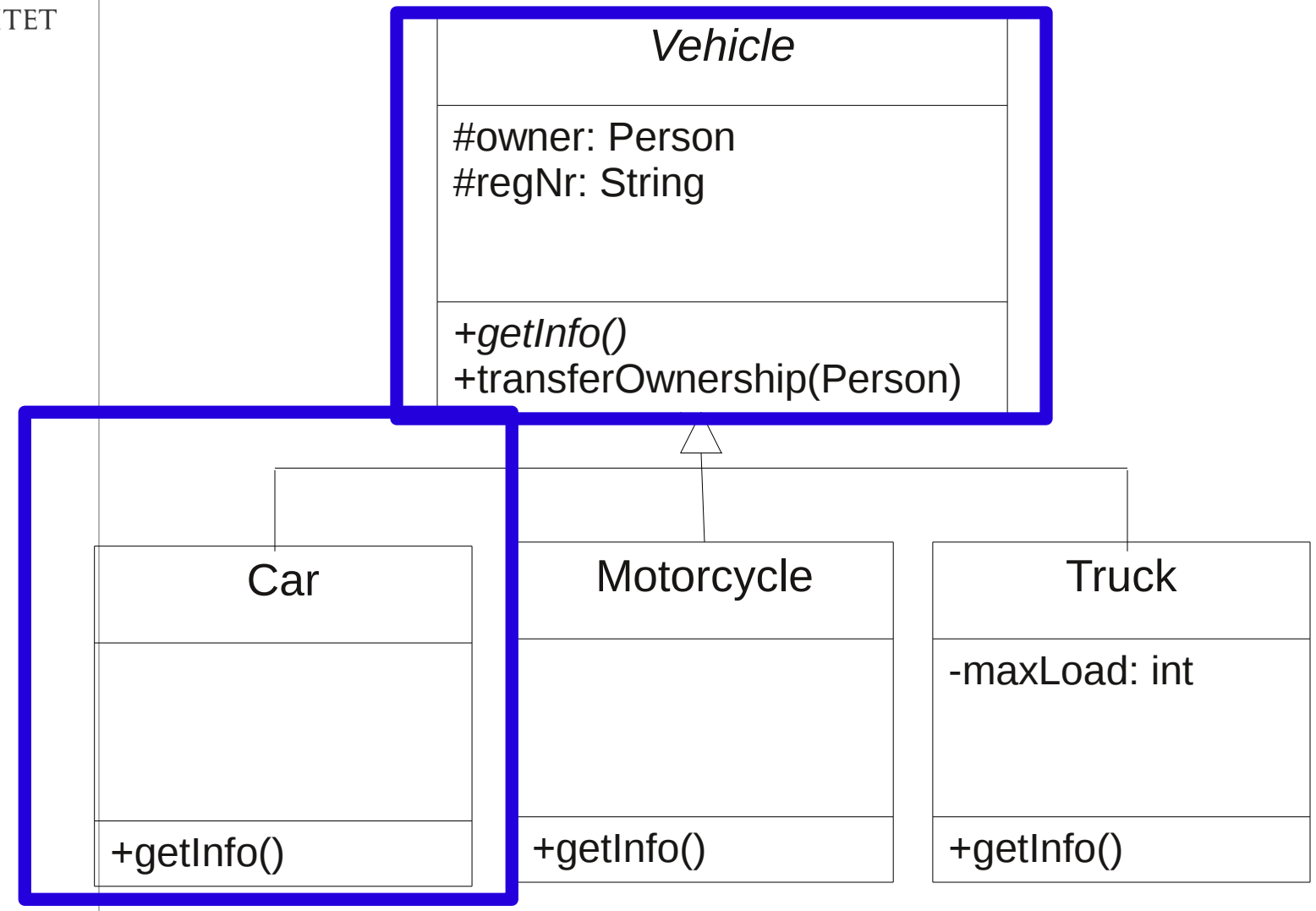
Arv – Vad ser vad?



Klassen Vehicle ser endast sig själv!

UPPSALA UNIVERSITET

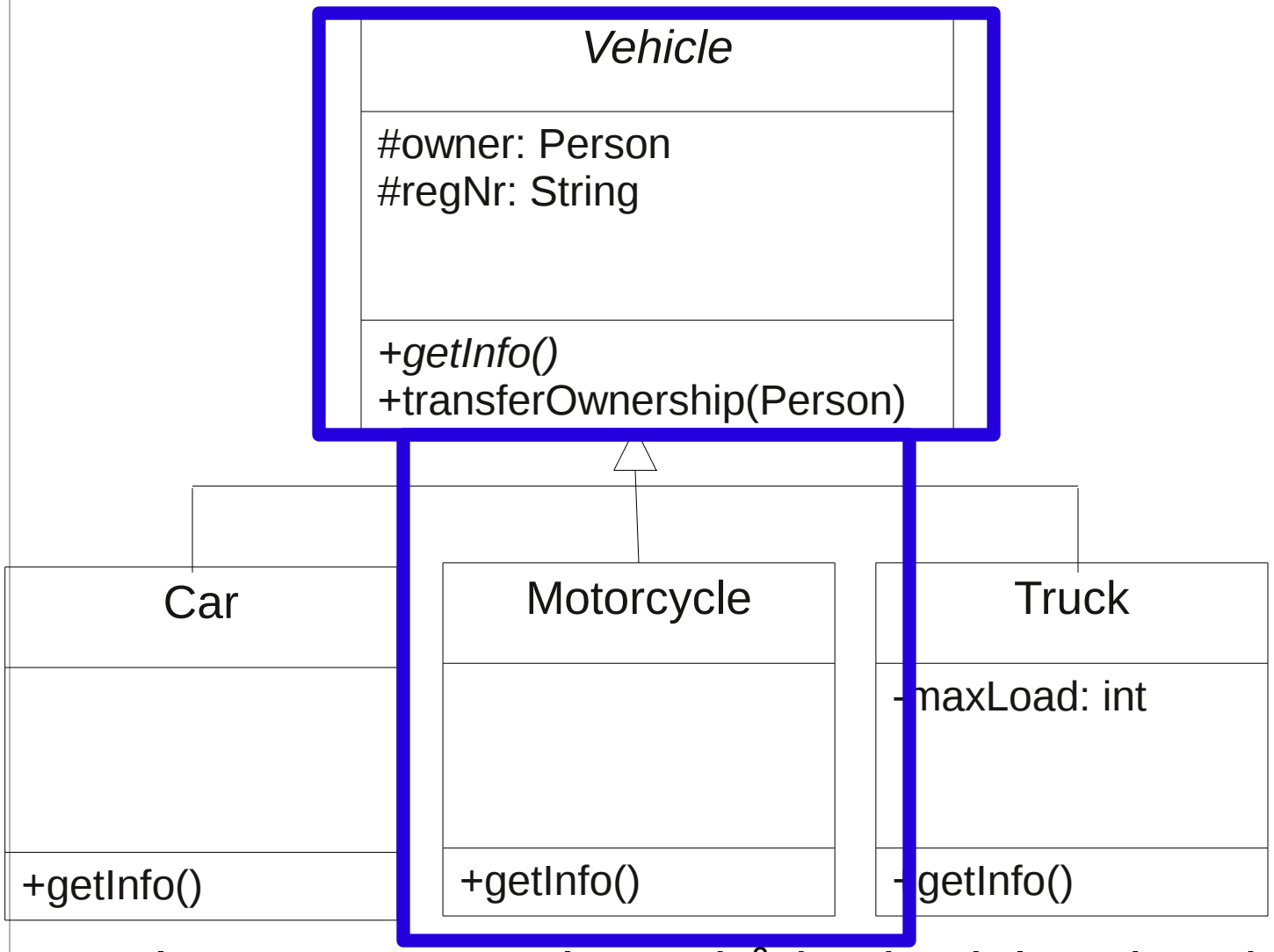
Arv – Vad ser vad?



Klassen Car ser både sig själv och Vehicle

UPPSALA UNIVERSITET

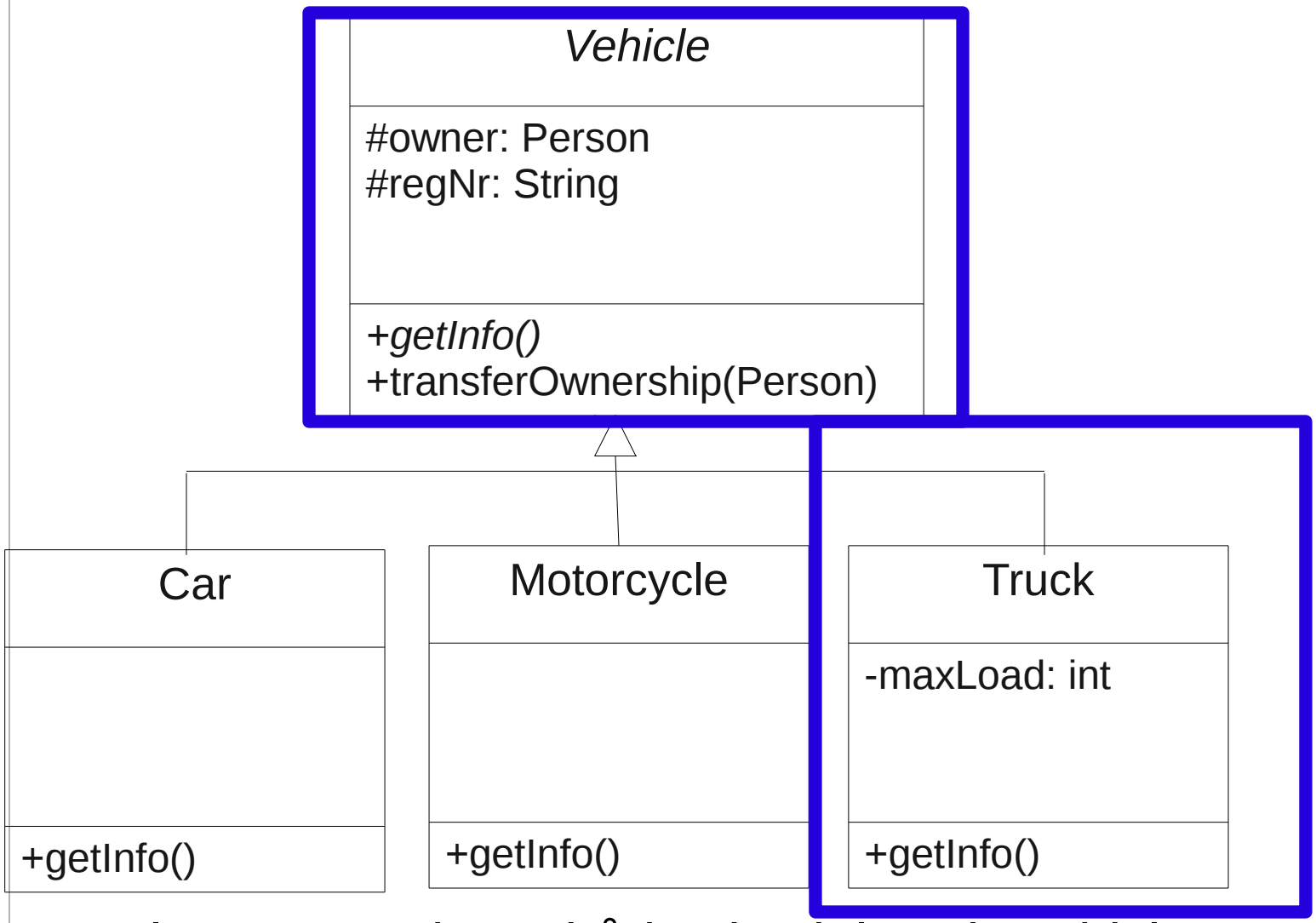
Arv – Vad ser vad?



Klassen Motorcycle ser både sig själv och Vehicle

UPPSALA UNIVERSITET

Arv – Vad ser vad?



Klassen Truck ser både sig själv och Vehicle

UPPSALA
UNIVERSITET

Arv - Fakta

En klass kan bara se sig själv och de klasser den ärvt från.

Exempel:

- Superklassen Vehicle kan bara se Vehicle

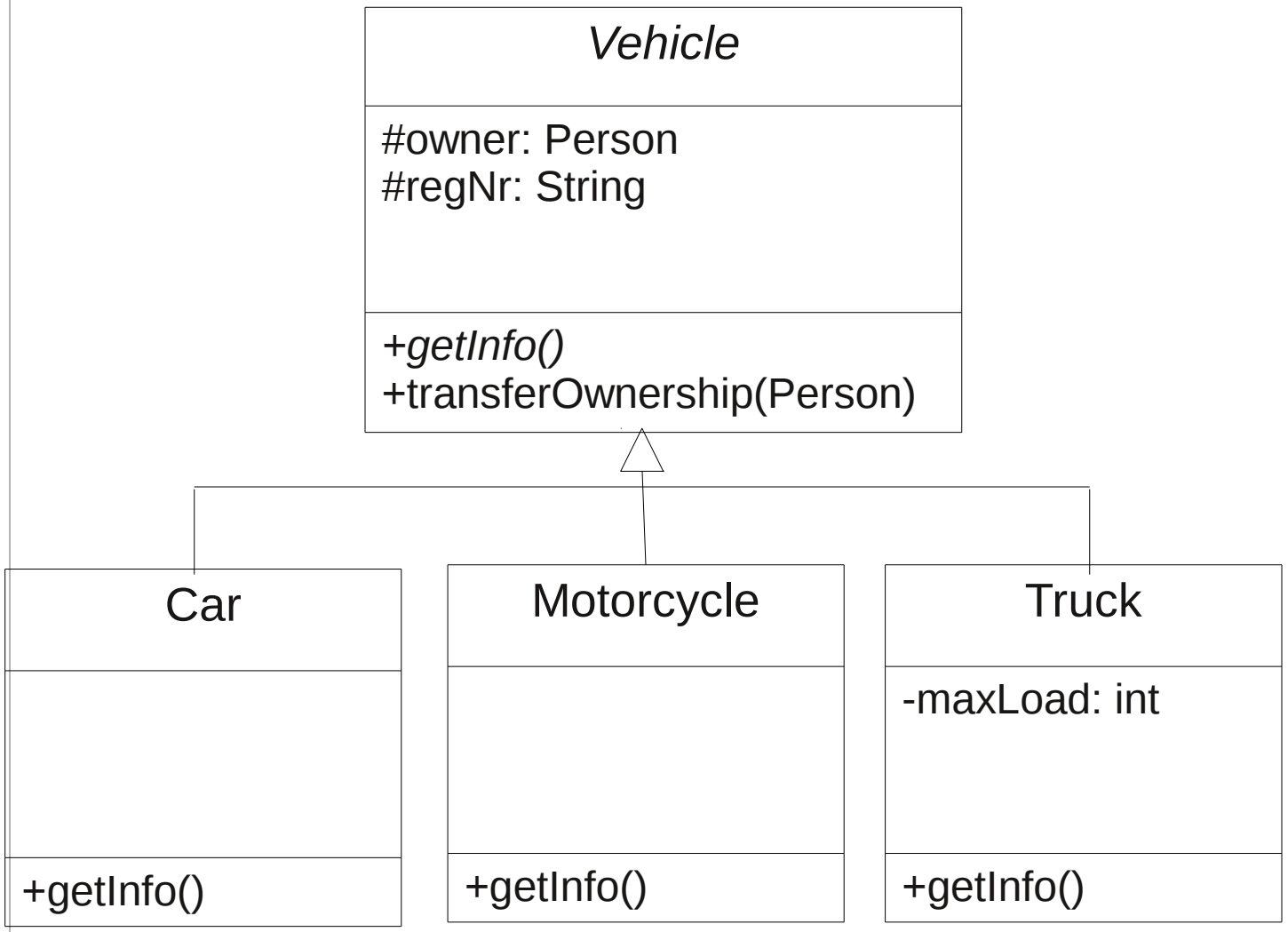
- Subklassen Car kan se både Car och superklassen Vehicle

- Subklassen Motorcycle kan se både Motorcycle och superklassen Vehicle

- Subklassen Truck kan se både Truck och superklassen Vehicle

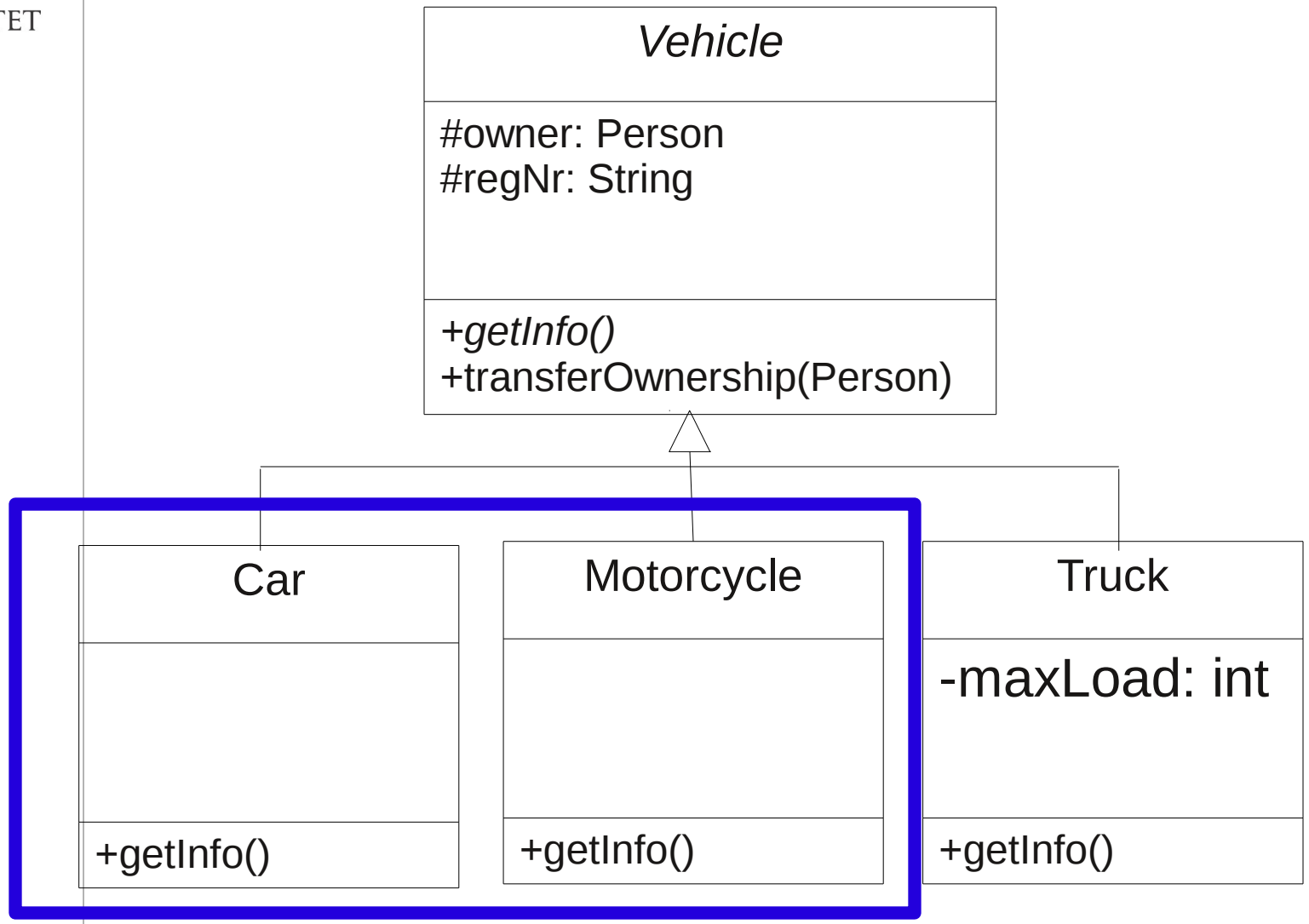
UPPSALA UNIVERSITET

Kan vi kategorisera klasserna ytterligare?



UPPSALA UNIVERSITET

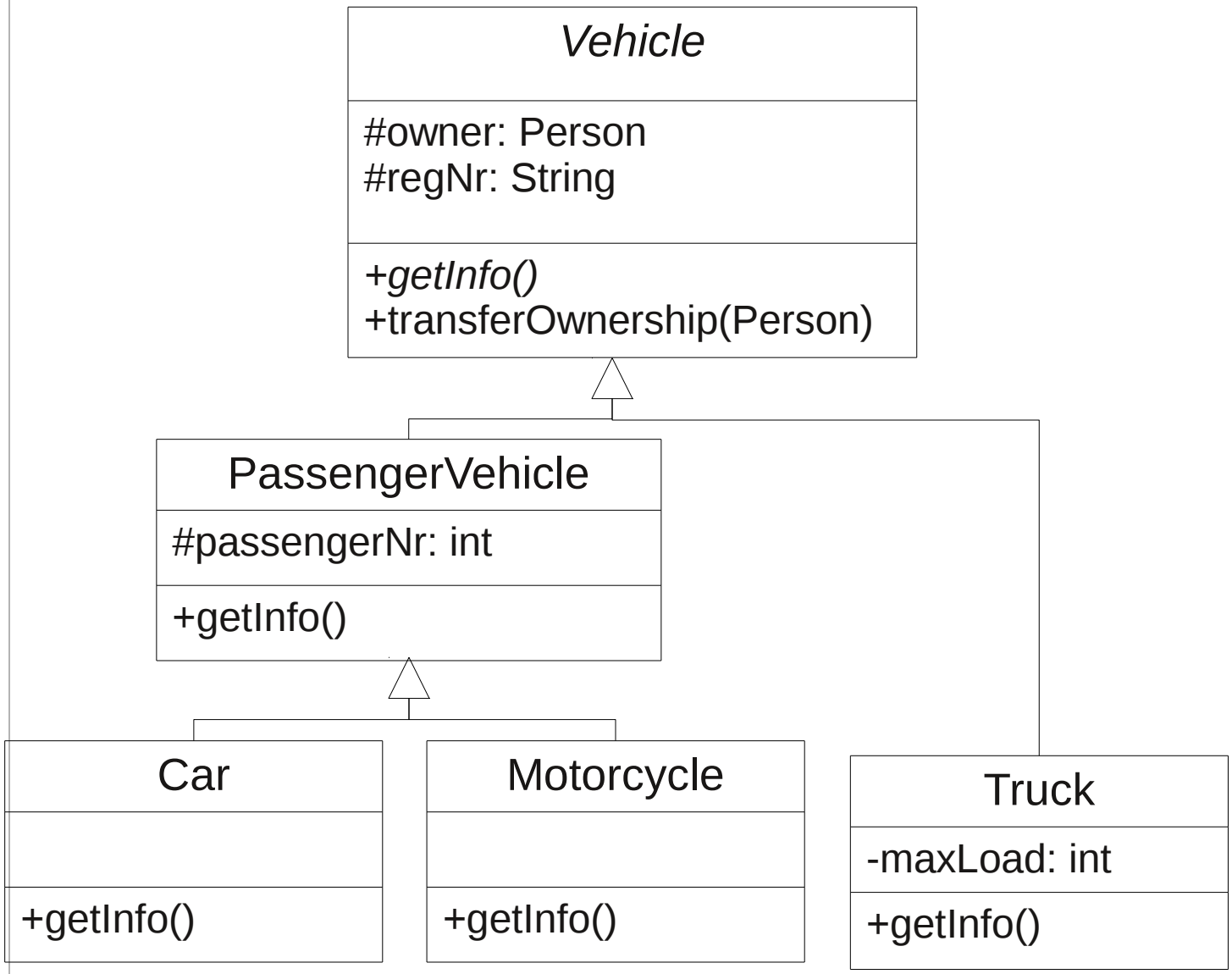
Kan vi kategorisera klasserna ytterligare?



Bilar och motorcyklar kan ha passagerare

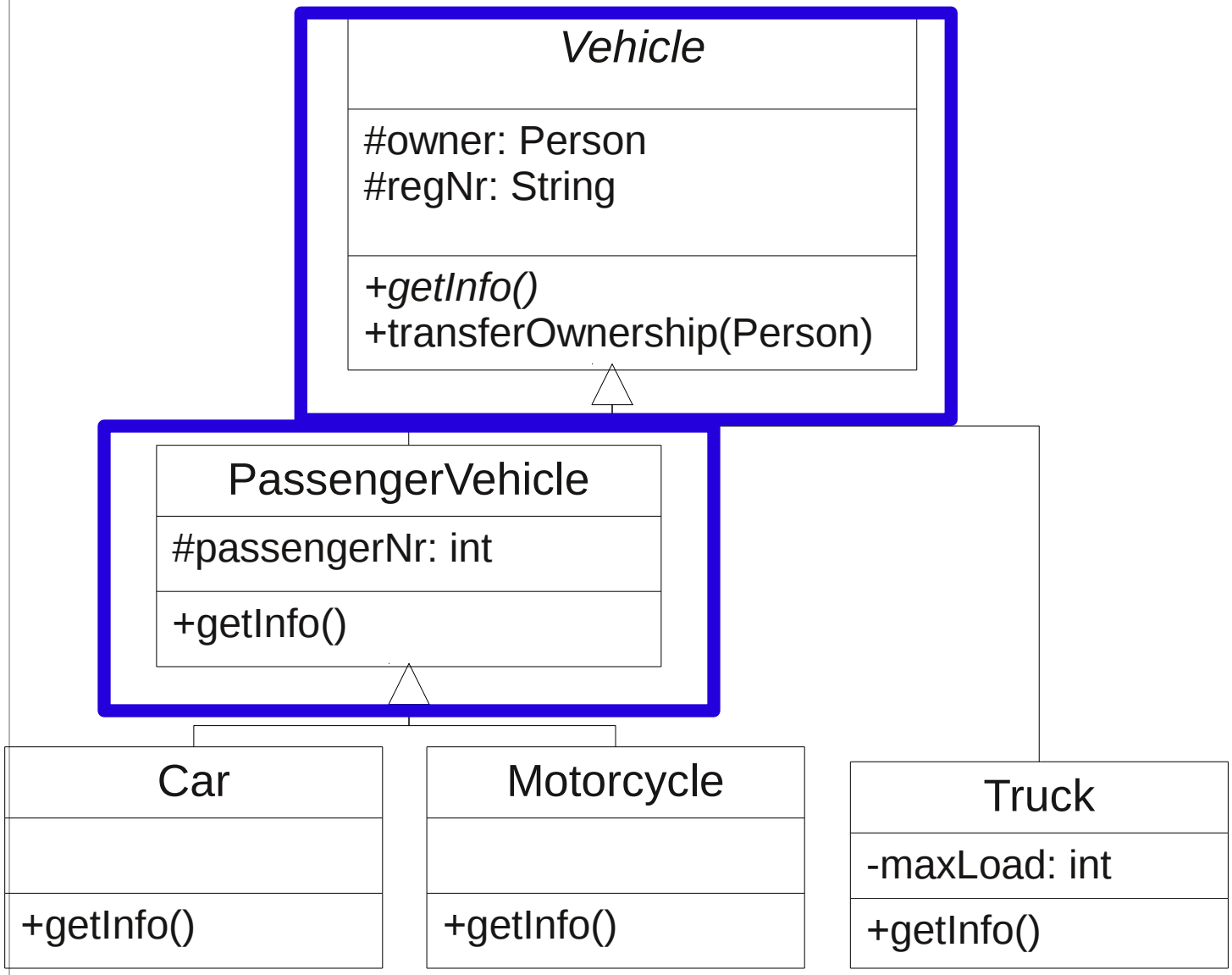
UPPSALA UNIVERSITET

Ny klass: passengerVehicle



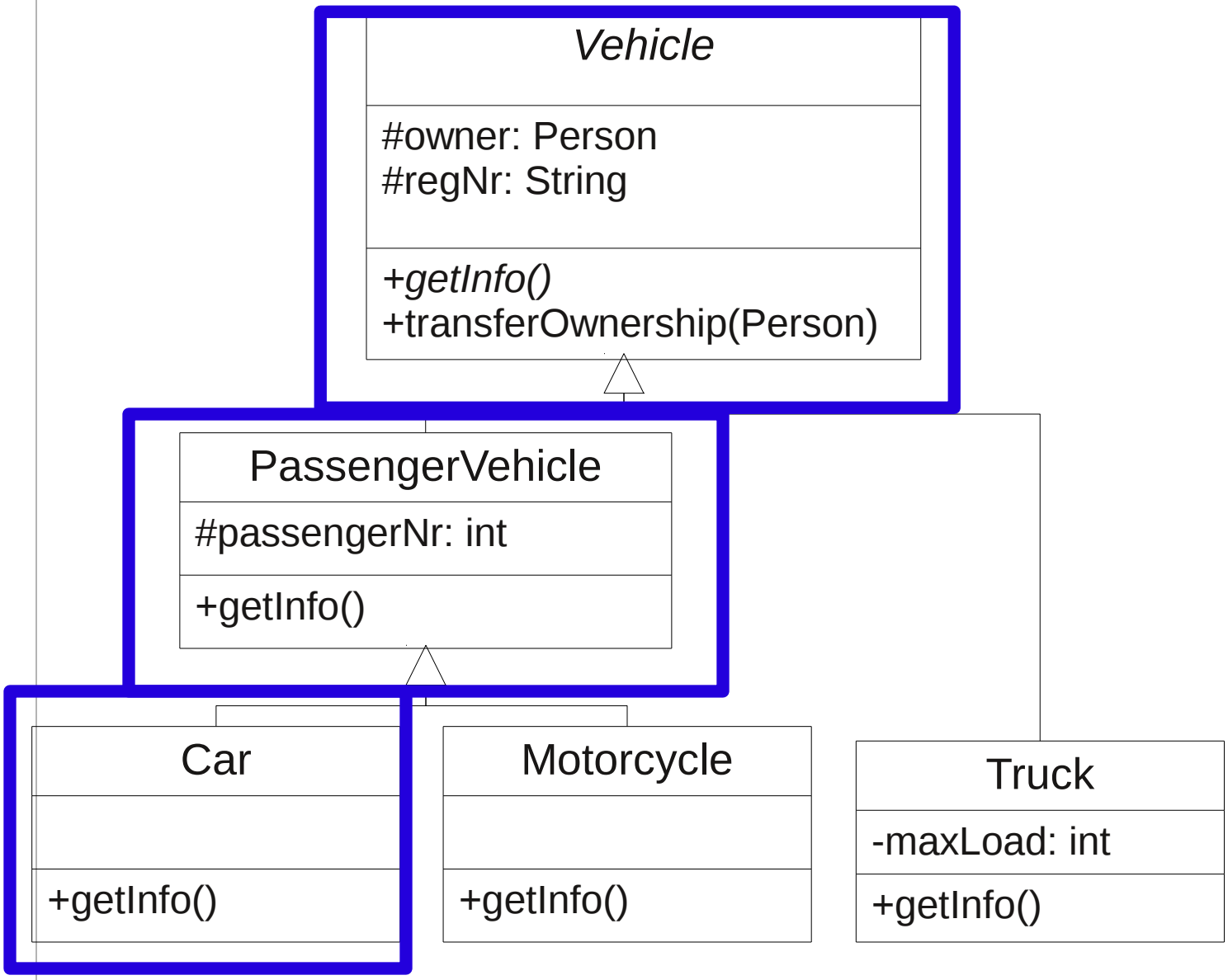
UPPSALA UNIVERSITET

Vad ser klassen PassengerVehicle?



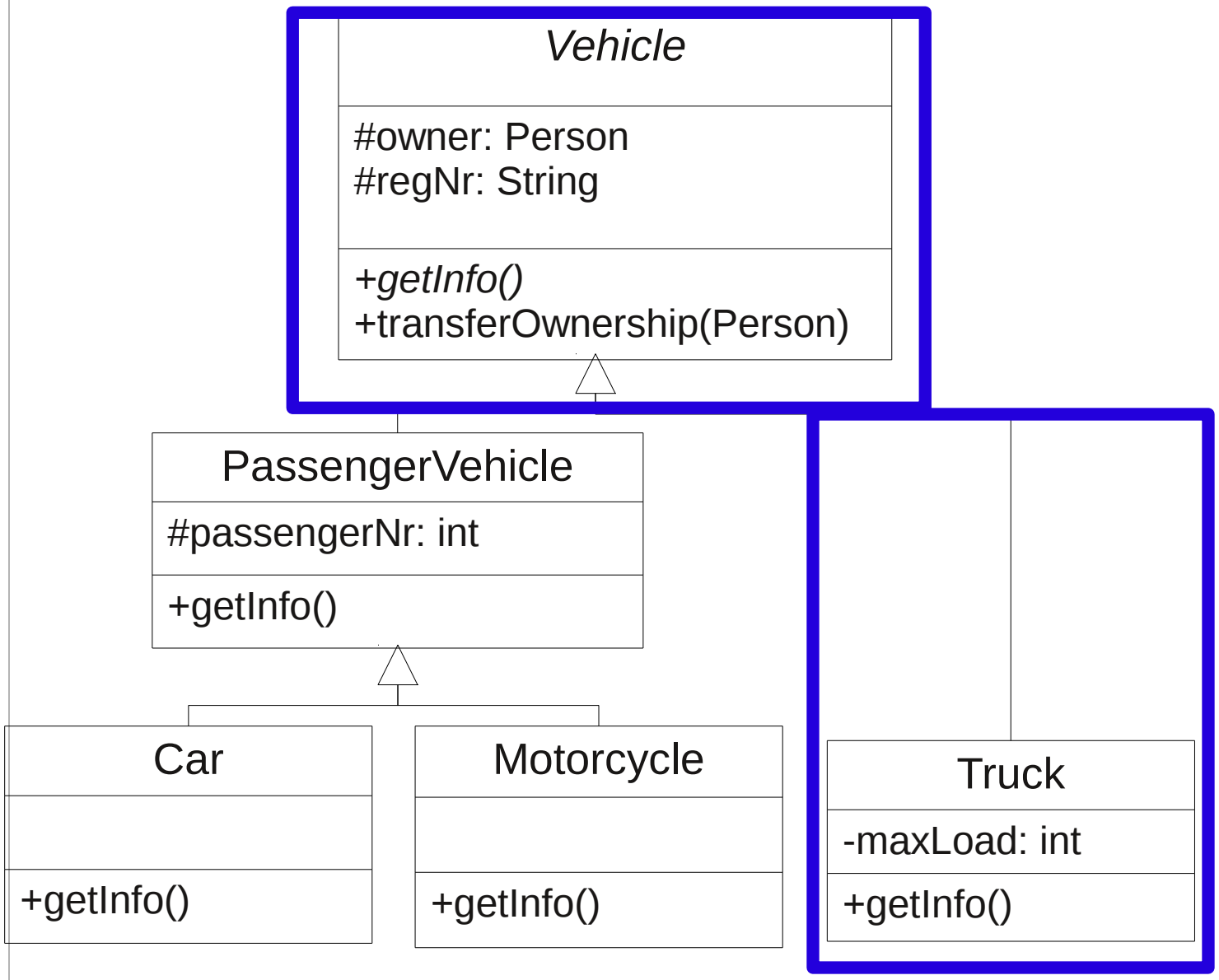
UPPSALA UNIVERSITET

Vad ser klassen Car?



UPPSALA UNIVERSITET

Vad ser klassen Truck?



UPPSALA
UNIVERSITET

Gammal kod

```
public class Car extends Vehicle{
  public Car(Person myOwner, String myReg){
    super(myOwner, myReg);
  }
}

public class Motorcycle extends Vehicle{
  public Motorcycle(Person myOwner, String myReg){
    super(myOwner, myReg);
  }
}
```

UPPSALA UNIVERSITET

Ny kod – konstruktorer

```
public class PassengerVehicle extends Vehicle{
  protected int passengerNr;

  public PassengerVehicle(Person myOwner,
                          String myReg, int p){
    super(myOwner, myName);
    passengerNr = p;
  }
}

public class Car extends PassengerVehicle {
  public Car(Person myOwner,
             String myReg, int p){
    super(myOwner, myName, p);
  }
}

public class Motorcycle extends PassengerVehicle {
  ...
```

UPPSALA UNIVERSITET

Ny kod - getInfo()

```
public class PassengerVehicle extends Vehicle{
   ...

   public String getInfo() {
      return myReg + “, max passagerare: “ +
             passengerNr;
   }
}

public class Car extends PassengerVehicle {
   ...

   public String getInfo() {
     return “Car: “+super.getInfo());
   }

}
```

UPPSALA UNIVERSITET

Använda klasserna

```
public class ExampleWithPassengers {

 public static void main(String args[]) {
   Person owner = new Person("Kalle", 20);
   ArrayList<Vehicle> vehicles =
                 new ArrayList<Vehicle>();
   vehicles.add(new Car(owner, "BBC123", 4));
   vehicles.add(new Motorcycle(owner,
                                  "MBC123", 1));
   vehicles.add(new Truck(owner, "TBC123", 100));

   System.out.println("My name is " +
                        owner.getName() +
                        "\nThese are my vehicles:");
   for (Vehicle v: vehicles) {
      System.out.println(v.getInfo());
   }
  }
}
```

UPPSALA
UNIVERSITET

Provkör!!! - Resultat

```
My name is Kalle
These are my vehicles:
Car: BBC123, max passagerare 4
Motorcycle: MBC123, max passagerare 1
Truck: TBC123, max load: 100
```

UPPSALA
UNIVERSITET

Lab 0

- Övning på objektorientering
- Implementera ett litet objektorienterat program om djur, steg för steg
- Frågor efter varje steg samt i slutet, som man bör kunna svara på
  - Frågorna även relevanta inför tentan
- Ingen redovisning, men materialet relevant för resten av kursen!
- Fråga gärna om ni är osäkra på något!

UPPSALA UNIVERSITET

Nästa vecka

- Tema:
  - Regulära uttryck
    - Fokus: hur de används i Java
  - Ändliga automater
  - Hashtabeller
  - Läsning/skrivning (repetition)
- Föreläsning
- Lab 1 – applicera ovanstående på språkteknologiska problem

UPPSALA UNIVERSITET

Jobba själv under veckan

- Gör klart lab 0
- Gör programmeringsövningar
  - Repetition, Eck kap 1-4
  - Objektorientering, Eck kap 5
- Repetera från tidigare kurser
  - Regulära uttryck och automater
  - Grundläggande programmering
- Läs på inför nästa vecka
  - Regulära uttryck (se hemsida för material!)
  - Hashtabeller, Eck kap 10.3-4