

Paveldējimas

Mindaugas Karpinskas
2017



Paveldėjimo pavyzdys

```
class A {  
    int i;  
}  
  
class B extends A {  
    int j;  
  
    int didinti() {  
        return i++;  
    }  
}
```

```
public class Paveldejimas1 {  
  
    public static void main(String[] args) {  
        B b = new B();  
        b.i = 50;  
        b.j = 60;  
        b.didinti();  
    }  
}
```

A Car

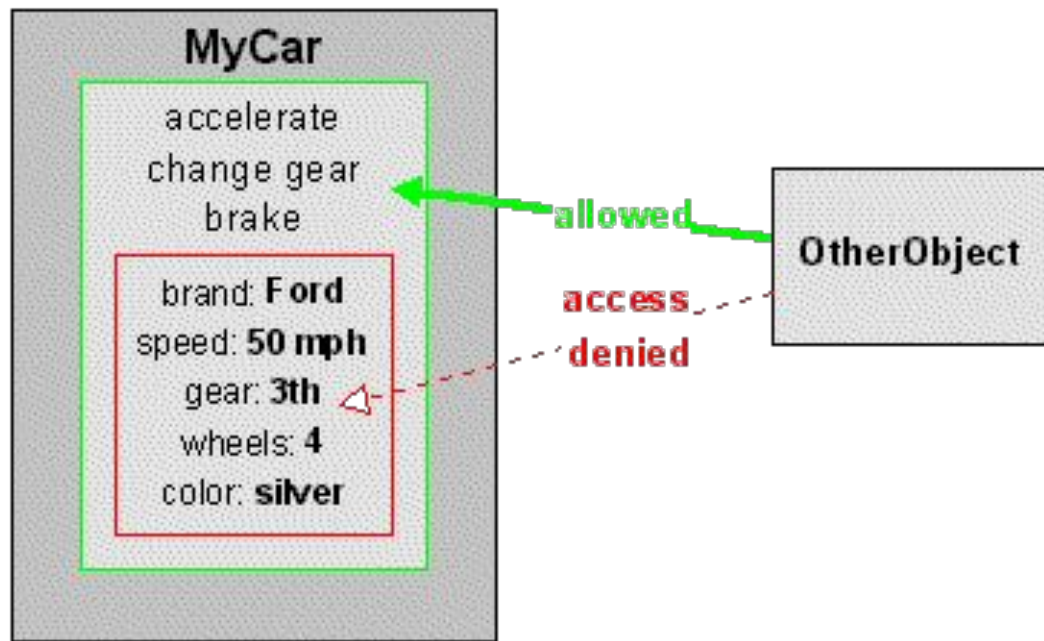
accelerate
change gear
brake

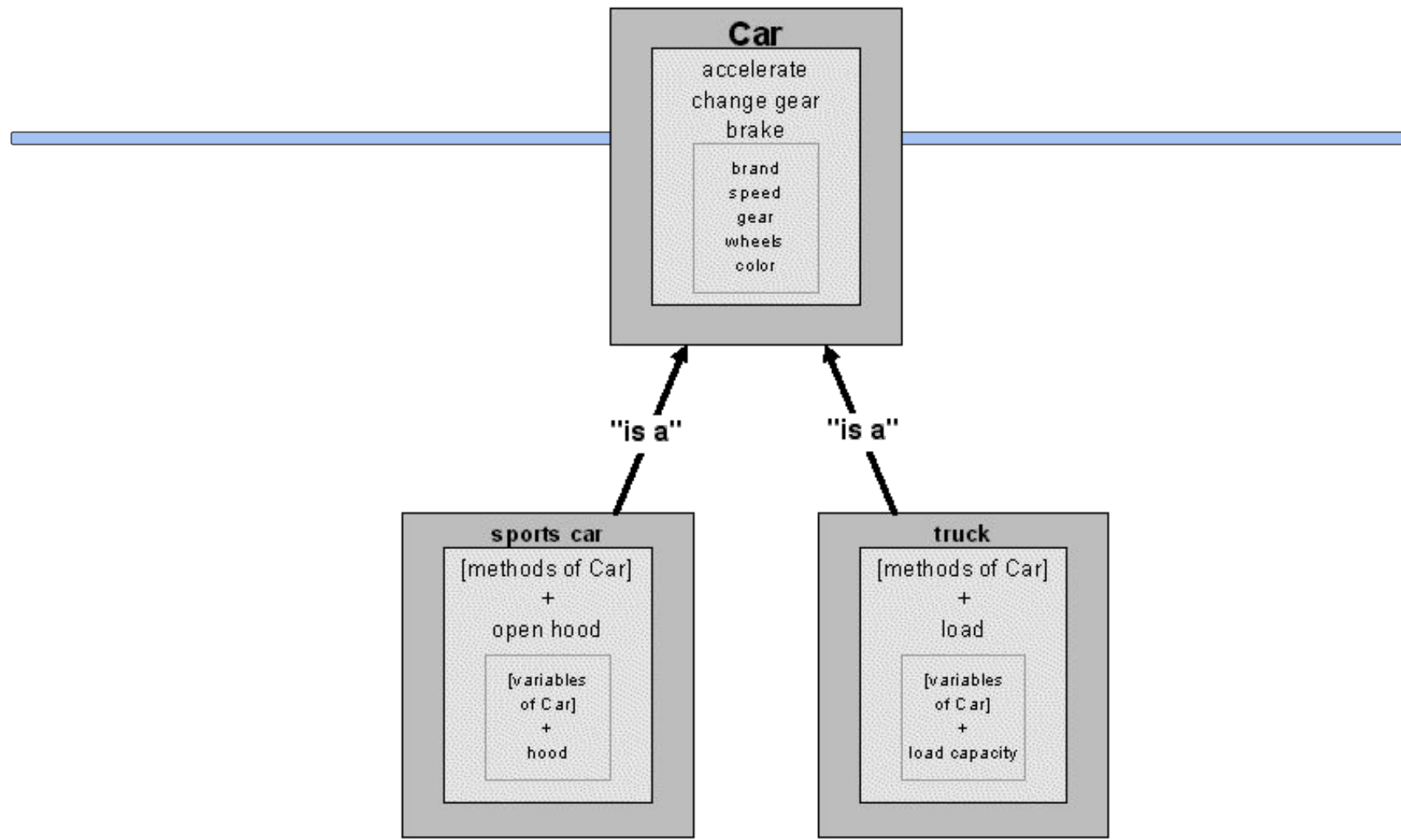
brand
speed
gear
wheels
color

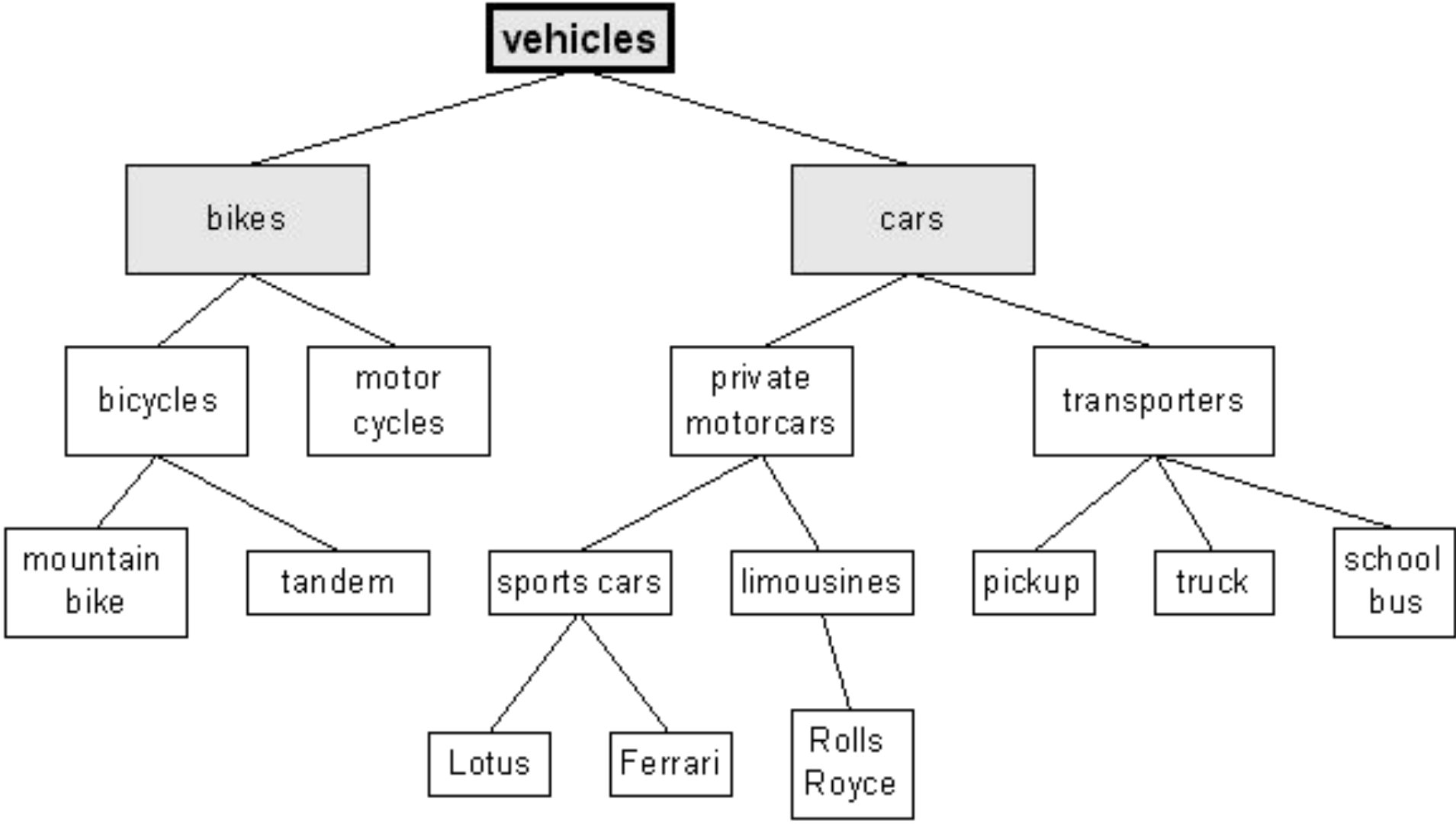
MyCar

accelerate
change gear
brake

brand: **Ford**
speed: **50 mph**
gear: **3th**
wheels: **4**
color: **silver**







Trumpai

Duomenų apsauga

Duomenų apgauba

Paveldimumas

Perrašomi metodai ir kintamieji

Links

1. https://www.learneroo.com/learn-java-programming?gclid=CjwKCAiA693RBR AwEiwALCc3uwtQQ5DvhSkls7wjKwjhrdhuJUOXaQ6TGNcyNlf7SMq1DXOjx 6oxHRoCqlgQAvD_BwE
2. https://www.tutorialspoint.com/java/java_inheritance.htm
3. <https://docs.oracle.com/javase/tutorial/java/landl/subclasses.html>
4. <https://beginnersbook.com/2013/03/inheritance-in-java/>
5. <https://www.javatpoint.com/inheritance-in-java>
6. <http://www.learnjavaonline.org/en/Inheritance>
7. <https://www.youtube.com/watch?v=9JpNY-XAseq>

Objektiniame programavime naudojami terminai

Terminas	Apibrėžimas
Paketas	Vienetas apjungiantis Java klases į vieną grupę.
Klasė	Duomenų tipo apibrėžimas, kuriame yra duomenys ir metodai (paprogramės).
Metodas	Javoje vartojamas vardas vadinti paprogramės egzempliorių.
Konstravimas	Programos vykdymo metu klasės kintamojo kūrimas.
Panaudojimo modifikatorius	Aprašo, kokia klasių aibė turi teisę keisti duotąjį klasės narį.

Pakartotinio kodo naudojimo būdai

- **Kodo kopijavimas ir taisymas*
- Kompozicija
- Paveldėjimas

Kompozicija

Esamų klasių objektų kūrimas naujoje klasėje

```
class A {  
    int i;  
}
```

```
class B {  
    int j;  
    A a;  
}
```

Klasėje B kintamasis a inicializuojamas reikšme null, todėl reikia sukurti klasės A objektą

1. Deklaravimo metu

```
class B {  
    int j;  
    A a = new A();  
}
```

2. Konstruktoriuje

```
class B {  
    int j;  
    A a;  
    B() {  
        a = new A();  
    }  
}
```

3. Prieš panaudojimą

```
class B {  
    int j;  
    A a;  
    int didinti() {  
        a = new A();  
        return a.i++;  
    }  
}
```

```
public class FinansaiVykdimas {
```

```
    private Pajamos pajamos;
```

```
    private Ivedimas ivedimas;
```

```
    private Meniu meniu;
```

```
    private void pajamuIvedimas() {  
        meniu.rodytiPajamuKategorijas();  
        int kategorija = ivedimas.intOf(7);  
        double suma = ivedimas.nextDouble();  
        pajamos.investi(kategorija, suma);  
    }
```

```
    private void pajamuPerziura() {  
        PajamuIrasas[] m = pajamos.visosPajamos();  
        for (int i = 0; i < m.length; i++) {  
            if (m[i] != null) {  
                String kategorija = String.valueOf(m[i].getKategorija());  
                System.out.println("suma: " + m[i].getSuma() + " kategorija: " +  
                                   kategorija);  
            }  
        }  
    }
```

Paveldimumas

- Siekiant išvengti kodo kartojimo, objektinėje paradigmoje įprasta naudoti paveldėjimą (angl. **inheritance**)
- Paveldėjimas leidžia sukurti naują klasę panaudojant ankstesnės klasės savybes
- Tėvinės klasės savybės būdingos ir vaicinei klasei
- Galima pridėti naujų savybių
- Paveldėjimui nusakyti naudojamas žodis **extends**

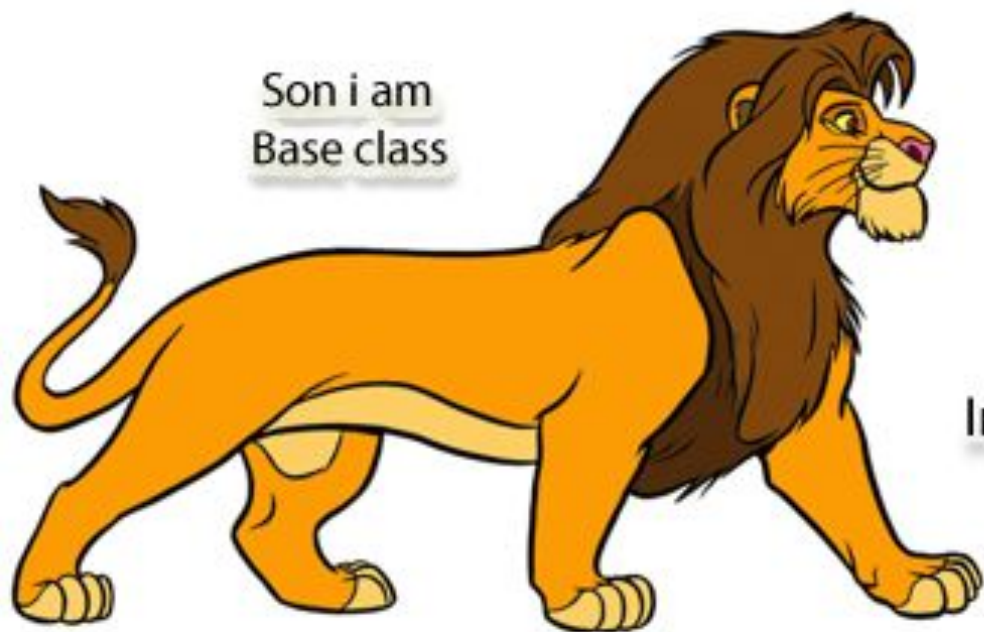
Tai priklausymo bendresnei klasei ("is a") ryšys.

- Kabrioletas yra Automobilio rūšis,
- Magistrantas - Studento atvejis,
- Ažuolas vienas iš Medžių.

Ryšys, kai viena klasė (tipas) yra kitos klasės poklasis (potipis), vadinamas paveldėjimo ryšiu. OP Naudojamos sąvokos:

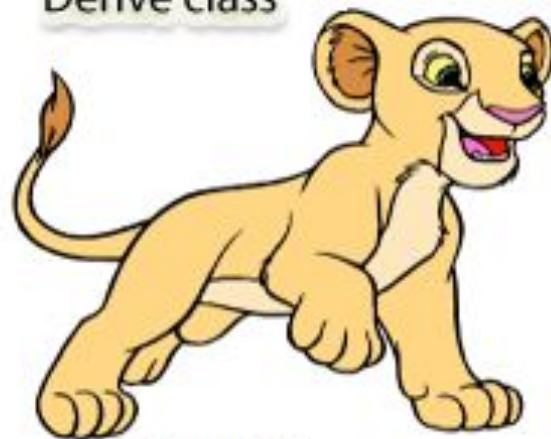
- Bazinė klasė <- Išvestinė klasė
- klasė-Tėvas <- klasė-Vaikas
- Superklasė <- Subklasė

Son i am
Base class



Inheritance

Dad i am
Derive class



Tutorial4us.com



```
class Convertible {  
  // Key (private)  
  // Speed : 155 (miles / hour)  
  // Weight 1600 kg  
  // Engine : 3.2 L S54 inline-6  
}
```



```
class Roadster extends Convertible {  
  // Speed : 165 (miles / hour)  
  // Weight 1399 kg  
}
```

Paveldėjimo pavyzdys

```
class A {  
    int i;  
}  
  
class B extends A {  
    int j;  
  
    int didinti() {  
        return i++;  
    }  
}
```

```
public class Paveldejimas1 {  
  
    public static void main(String[] args) {  
        B b = new B();  
        b.i = 50;  
        b.j = 60;  
        b.didinti();  
    }  
}
```

Užduotis0 Paveldėjimas

Reikalingos dvi klasės: **Tervas**, **Vaikas**. **Vaikas** paveldi **Tervas** klau.

Tervas - turi metodą “public String **pirmas()**” kuris grąžina tekstą “*Pirma klasė*”

Vaikas - turi metodą “public String **antras()**” kuris grąžina tekstą “*Antra klasė*”

Trečioje klasėje main metode susikurkime du kintamuosius: **Tervas** ir **Vaikas** tipo.

Pabandykime iškviesti abu metodus ir grąžintą rezultatą išspausdinti.

E->D

Konstruktoriai ir paveldimumas

- Konstruktoriai nėra paveldimi
- Tėvinės klasės kintamieji turi būti inicijuoti vaikinėje klasėje
- Galima kviesti tėvinės klasės konstruktorių iš vaikinės klasės konstruktoriaus
- Kuriant vaikinių klasių objektus, yra tokia veiksmų seka:
 - Išskiriama vieta
 - Kintamieji inicializuojami pradžioje tėvinės klasės, paskui vaikinės
 - *Konstruktorius kviečiamas pradžioje tėvinės klasės, paskui vaikinės*****

Konstruktoriai

- Tėvinės klasės konstruktorius kviečiamas su **super()** nurodant parametrus
- Kuriant objektus, pradžioje kviečiamas tėvinės klasės konstruktorius, o tik tada vaikinės klasės konstruktorius.
- Taip būna net tada, jei programoje neparašyta, kad reikia kviesti tėvinės klasės konstruktorių
- Kviečiant tėvinės klasės konstruktorių, negali būti jokio kito žingsnio vaikinės klasės konstruktoriuje prieš šį žingsnį
- Tos pačios klasės konstruktorius kviečiamas panaudojant **this()**

Konstruktoriai. 1 pavyzdys

```
class A {  
    A() {  
        System.out.println("Kuriam A");  
    }  
}  
class B extends A {  
    B() {  
        System.out.println("Kuriam B");  
    }  
}  
class C extends B {  
    C() {  
        System.out.println("Kuriam C");  
    }  
}
```

```
public static void main(String[] args)  
{  
    C c = new C();  
}
```



Konstruktoriai. 1 pavyzdys

```
class A {  
    A() {  
        System.out.println("Kuriam A");  
    }  
}  
class B extends A {  
    B() {  
        System.out.println("Kuriam B");  
    }  
}  
class C extends B {  
    C() {  
        System.out.println("Kuriam C");  
    }  
}
```

```
public static void main(String[] args) {  
    C c = new C();  
}
```

Kuriam A
Kuriam B
Kuriam C

Konstruktoriai. 1 pavyzdys (default constructor)

```
class A {  
    A() {  
        System.out.println("Kuriam A");  
    }  
}  
class B extends A {  
    B() {  
        super();  
        System.out.println("Kuriam B");  
    }  
}  
class C extends B {  
    C() {  
        super();  
        System.out.println("Kuriam C");  
    }  
}
```

```
public static void main(String[] args)  
{  
    C c = new C();  
}
```

Kuriam A
Kuriam B
Kuriam C

Konstruktoriai. 2 pavyzdys

```
class A {  
    int i;  
    A() {  
        i = 5;  
    }  
}  
class B extends A {  
    int j;  
    B() {  
        // šioje vietoje kviečiamas klasės A  
        konstruktorius  
        j = i + 1;  
    }  
}
```

```
public static void main(String[] args) {  
    B b = new B();  
    System.out.println("i = " + b.i);  
    System.out.println("j = " + b.j);  
}
```

Konstruktoriai. 2 pavyzdys

```
class A {  
    int i;  
    A() {  
        i = 5;  
    }  
}  
class B extends A {  
    int j;  
    B() {  
        // šioje vietoje kviečiamas klasės A  
        konstruktorius  
        j = i + 1;  
    }  
}
```

```
public static void main(String[] args) {  
    B b = new B();  
    System.out.println("i = " + b.i);  
    System.out.println("j = " + b.j);  
}
```

```
i = 5  
j = 6
```

Konstruktoriai. 3 pavyzdys (su argumentu)

```
class A {  
    int i;  
    A(int k) {  
        i = k;  
    }  
}  
  
class B extends A {  
    int j;  
    B(int m) {  
        // čia nieko įterpti negalima  
        super(m); // būtinas  
        j = i + 1;  
    }  
}
```

```
public static void main(String[] args) {  
    B b = new B(10);  
    System.out.println("i = " + b.i);  
    System.out.println("j = " + b.j);  
}
```

Konstruktoriai. 3 pavyzdys (su argumentu)

```
class A {  
    int i;  
    A(int k) {  
        i = k;  
    }  
}  
  
class B extends A {  
    int j;  
    B(int m) {  
        // čia nieko įterpti negalima  
        super(m); // būtinas  
        j = i + 1;  
    }  
}
```

```
public static void main(String[] args) {  
    B b = new B(10);  
    System.out.println("i = " + b.i);  
    System.out.println("j = " + b.j);  
}
```

```
i = 10  
j = 11
```

Užduotis Paveldėjimas1

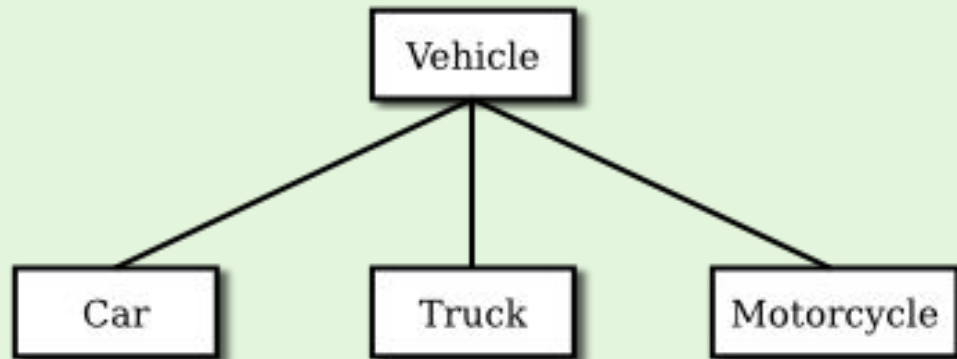
Sukurti trijų lygių klasių hierarchiją: Pirmas, Antras, Trecias. Visos klasės turi turėti po du konstruktorius: default'ini ir su parametru/-ais. Keikvienas konstruktorius turi informuoti išvesdamas tekstą į konsolę, apie savo kurimą, jei konstruktorius su parametru, jis turi išvesti ir jo reikšmę.

Main metode sukurti kelis egzempliorius klasiu: Pirmas, Antras, Trečias.

E

Užduotis Paveldėjimas2

- Klasės: TransportoPriemone, Masina, Sunkvežimis, Motociklas
 - Bendri duomenys: ratų sk., markė, spalva
 - Bendri metodai: judek(), sustok(), iKaire(), iDesine()
-
- Kiekviena vaikinė klasė turi pridėti
 - bent du naujus laukus
 - bent du naujus metodus



Užduotis Paveldėjimas3

- Klasė Asmuo turi:
 - Du private laukus: vardas, pavadrė; reikšmės nekeičiamos ir privalomos;
 - du viešus metodus kurie grąžina laukų reikšmes;
 - kuriant klasės egzempliorius/objektus būtina reikia perduoti vardą, pavardę
 - metodą informacija(); išspausdina informaciją apie asmenį.
- Klasė Studentas:
 - praplečia Asmuo klasę lauku: studento numeris;
 - turi metodą kuris:
 - grąžina studento numerį;
 - pakeičia studento numeri;
 - studento objektas gali būti sukurtas su ir be studento numerio
 - turi metodą kuris išspausdina asmens informaciją ir stud. nr.
- Klasių laukai turi būti nepasiekiami kitiems klasėms;

Main metode sukurti keleta asmens, studento egzempliorių su skirtinga informacija ir iškviesti info* metodus.

Perrašomi metodai ir kintamieji

- Metodo perrašymas (angl. overriding) - tai metodo kūno (angl. body) perrašymas vaikinėje klasėje
- Perrašyti metodai ir kintamieji turi tokį pat aprašymą kaip tėvinėje klasėje
- Vaikinėje klasėje abi versijos yra pasiekiamos
- Prie tėvinės klasės kintamojo arba metodo galima prieiti panaudojant **super**
- Perrašant metodus, jų priėjimo teisių negalima sumažinti

Metodo perrašymo pavyzdys

```
class A {  
    int i = 5;  
  
    int didinti() {  
        return i + 1;  
    }  
}  
class B extends A {  
    int didinti() {  
        return i + 2;  
    }  
    void spausdinkAbu() {  
        System.out.println(didinti());  
        System.out.println(super.didinti());  
    }  
}
```

```
public static void main(String[] args) {  
    B b = new B();  
    b.spausdinkAbu();  
}
```

?

Metodo perrašymo pavyzdys


```
class A {  
    int i = 5;  
  
    int didinti() {  
        return i + 1;  
    }  
}  
class B extends A {  
    int didinti() {  
        return i + 2;  
    }  
    void spausdinkAbu() {  
        System.out.println(didinti());  
        System.out.println(super.didinti())  
    }  
};
```

```
public static void main(String[] args) {  
    B b = new B();  
    b.spausdinkAbu();  
}
```

7
6

Perrašymas vs. perkrovimas

```
class A {  
    int metodos() {  
        return 5;  
    }  
    // perkrovimas (angl. overloading)  
    int metodos(int k) {  
        return k + 1;  
    }  
}  
class B extends A {  
    // perrašymas (angl. overriding)  
    int metodos() {  
        return 6;  
    }  
}
```



Įsiminti !!!

Užduotis Paveldėjimas4

Užduoties Paveldėjimas3 klasė studentas turi perrašyti metoda informacija(); papildant studento informacija.

*Asmens laukai turi but private...

?

;

Parašykite klasę, turinčią kintamąjį ir metodą. Parašykite kitą klasę paveldėtą iš pirmosios. Parodykite, kad paveldėtoje klasėje pasiekiami bazinės klasės metodai ir kintamieji. Parodykite, kad bazinės klasės konstruktorius paveldėtos klasės konstruktoriuje kviečiamas visada ir kad bazinės klasės konstruktorius kviečiamas prieš kviečiant kitą paveldėtos klasės konstruktorių.

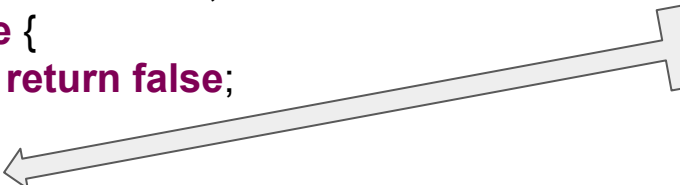
Užduotis

Parašykite klasę, turinčią kintamąjį ir metodą. Parašykite kitą klasę paveldėtą iš pirmosios. Parodykite, kad paveldėtoje klasėje pasiekiami bazinės klasės metodai ir kintamieji. Parodykite, kad bazinės klasės konstruktorius paveldėtos klasės konstruktoriuje kviečiamas visada ir kad bazinės klasės konstruktorius kviečiamas prieš kviečiant kitą paveldėtos klasės konstruktorių.

```
public class ClassTask {  
    public static void main(String[] args) {  
        B b = new B();  
        b.didint();  
        System.out.println(b.i);  
        b.spausdinti();  
    }  
}  
  
class A {  
    int i = 10;  
    void didint() {  
        i++;  
    }  
}  
  
class B extends A {  
    void spausdinti() {  
        System.out.println(i);  
    }  
}
```


Pvz.: ReplaceChars

```
class ReplaceChars {  
    private char myArray[];  
    private int curPos = 0;  
    public ReplaceChars(char someArray[]) {  
        myArray = someArray;  
    }  
    public boolean replaceNextChar(char c, char d) {  
        if (newPositionSet(c)) {  
            myArray[curPos] = d;  
            return true;  
        } else {  
            return false;  
        }  
    }  
    .....  
}
```



```
private boolean newPositionSet(char c) {  
    int i = curPos;  
    while (i < myArray.length) {  
        if (c == myArray[i]) {  
            curPos = i;  
            return true;  
        } else {  
            curPos = i;  
            i++;  
        }  
    }  
    return false;  
}  
public boolean atEnd() {  
    return (curPos == myArray.length);  
}
```

Pvz.: ReplaceChars

- **newPositionSet** metodas pažymėtas private. Mes apsaugojame vietoje kintamojo metodą. Tokia metodo apsauga garantuoja, kad kiti objektai negalės pasinaudoti newPositionSet metodu ir todėl niekas nesugadins einamojo indekso reikšmės.
- Nėra galimybės pakeisti myArray[] reikšmės, todėl jos negali sugadinti kitos procedūros.
- Ta pati pastaba galioja ir einamojo masyvo elemento indeksui: **curPos**
- Java kalboje reikalingą kintamąjį ir metodą apgaubėme vienoje klasėje ir tai padėjo išspręsti informacijos paslėpimo problemą.
- *Java privalumai dar labiau išryškėja, kai sprendžiame uždavinį, kaip atlikti aprašytą simbolių pakeitimo operaciją vienu metu keliuose masyvuose. Java atveju tiesiog sukuriame keletą klasės egzempliorių ir su jais toliau dirbame*

Pvz.: ReplaceChars

```
public static void main(String[] args) {  
  
    ReplaceChars solution1 = new ReplaceChars("Java galinga!".toCharArray());  
    ReplaceChars solution2 = new ReplaceChars("Aš išmoksiu Java!".toCharArray());  
  
    while (!solution1.atEnd()) {  
        solution1.replaceNextChar('a', 'x');  
    }  
    while (!solution2.atEnd()) {  
        solution2.replaceNextChar('o', 'y');  
    }  
}
```

Java klasės metodus mes automatiškai gauname, kad einamoji pozicija neišeis iš masyvo ribų.

Paveldėjimo pavyzdys

Anksčiau pateiktame pavyzdyje ReplaceChars pakeisdavo pirmą sutiktą c simboliu nauju simboliu d. Tarkime, dabar sugalvojome pakeisti ne tik pirmą sutiktą, bet turėti metodą, kuris keičia visus c simbolius simboliu d.

Paveldimumas, kaip tik čia ir praverčia, nes mes praplečiame anksčiau parašytą klasę nekeisdami jos pradinio varianto. Todėl tiems, kuriems bus reikalingas supaprastintas variantas naudosis pradiniu klasės variantu, o jei kas norės papildyti savais metodais - rašys savą klasės praplėtimo variantą. Kadangi klasės plėtiniai turi naujus vardus, nekils problemos dėl vardo dubliavimosi. Dabar pateiksime savo praplėtimo variantą:

BetterReplaceNextChar

```
class BetterReplaceNextChar extends ReplaceChars {  
  
    public BetterReplaceNextChar(char[] someArray) {  
        super(someArray);  
    }  
  
    public int replaceAllChars(char c, char d) {  
        int i = 0;  
        while (!atEnd()) {  
            replaceNextChar(c, d);  
            i++;  
        }  
        return i;  
    }  
}
```

Taigi dabar turime naują klasę BetterReplaceNextChar, kuri paveldi ankstesniosios ReplaceChars visus kintamuosius ir metodus ir turi naują metodą.

?

ND

- Sukurti:
 - klasę TransportoPriemone(ratuSkaicius, spalva)
 - klasę Automobilis, kuri paveldi TransportoPriemone su laukas: kuroTipas(panaudojant **enum**), variklioTuris, cilindruSkaicius.
 - klasę dviratis(pavaruSkaicius) paveldiTransportoPriemone
 - klasę Sunkvezimis, kuri paveldi Automobilis su laukais: krovinioTalpa, krovinioSvoris
- Metodas TransportoPriemone.spausdinti() išspausdina informaciją apie (ratuSkaicius, spalva)
- Kitos vaikinės klasės turi perrašyti metodą, papildant informaciją apie transporto priemonę
- TransportoPriemone konstruktoriuje pakeičiame ratuSkaicius=0, spalva=nežinoma
 - Kitos vaikinės klasės privalo konstruktoriuje priskirti pradinės reikšmės savo klasės kintamiesiems ir jei galima, perrašyti tevinės klasės kintamųjų informaciją