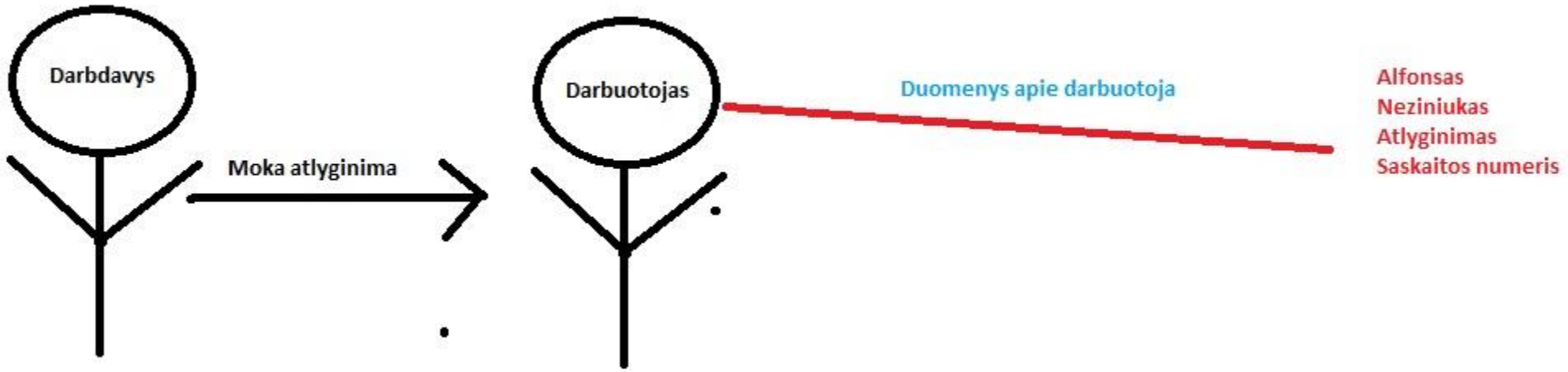
Java programavimo kalba

Java klasės laukai ir metodai.

Abstrakčios klasės

Objektas ir klases

- Uzduotis – Darbdavys nori sumoketi atlyginima darbuotojui pagal pasirasyta sutarti.
- Loginis sprendimas – mums reikia darbdavio kuris moketu, mums reikia darbuotojo kuriam sumoketume.



Objektas ir klases

- Darbdavys siuo atveju esame mes.
  Jeigu as gaunu uzklausa is sekretores, kad darbuotojas nori gauti atlyginima, as nutariu jam ji ir sumoketi (isimtiniais atvejais bandysiu nusukti kazkaip)

- Tam kad as sumokeciau darbuotojui atlyginima man kyla klausimas o kuriam darbuotojui reikia sumoketi ? Tam kad suzinociau man reiketu kad paskytu bent jau varda/pavarde. Tam kad zinociau kur sumoketi reiketu kad pasakytu savo banko saskaitos numeri. Taip pat noreciau zinoti ir kiek atlyginimo as visgi jam moku.

- Kadangi darbdavys esu as man nebutina pasakyti kas as esu, as ir taip zinau. Todel kai mes sumokesime atlyginima, mes pranesime sekretorei, jog sumokejome.

Objektas ir klases

- Programavimo sprendimas:
  1. Atspausdinsime rezultata (pasakysime kad ruosiames kazka nuveikti).
  2. Sukursime **klase** darbuotojas, kuris tures laukus – vardas/pavarde ir t.t.
  3. Sukursime darbuotoja (**objekta**) “Alfonsas Neziniukas 1500 LT213544654564654”.
  4. Sumokesime Alfonsui 1500 atlyginima i nurodyta saskaita.
  5. Pranesime sekretorei, jog sumokejome.

Priskyrimas pagal reikšmę (assing to variable)

Jei kintamasis, kuris yra primityvaus tipo (skaičiai, boolean, char), priskiriamas kitam kintamajam tai jo reikšmė kopijuojama.

int a = 100;

int b = a;

a = 200;

Kokia b reikšmė?

Parametrų perdavimas pagal reikšmę (pass parameter to a method)

Jei kintamasis, kuris yra primityvaus tipo, perduodamas į metodą kaip parametras, tai į metodą persiduoda jo reikšmė:

```
void metodas(int a) {
        a = 100;
}

void kitas() {
        int a = 200;
        this.metodas(a);
        System.out.print(a);
}
```

Kokią a reikšmę atspausdins?

Priskyrimas pagal nuorodą (a pointer to instance)

Jei kintamasis yra String tipo tai kopijuojama nuoroda į reikšmę. Bet kadangi String tipo reikšmės yra **nemodifikuojamos (!!!)**, tad keičiant eilutę bus sukuriama nauja:

String a = "Labas";

String b = a;

a = a + " vakaras";

Kokia b reikšmė?

Priskyrimas pagal nuorodą (a pointer to instance)

Jei kintamasis, kuris nėra primityvaus tipo (masyvas ar objektas), priskiriamas kitam kintamajam kopijuojama ne reikšmė bet nuoroda į reikšmę:

```
int[] a = {1, 10, 100};
int[] b = a;
a[0] = 2;
```

Kokia b[0] reikšmė?

Priskyrimas pagal nuorodą (a pointer to instance)

Jei kintamasis, kuris nėra primityvaus tipo, perduodamas į metodą kaip parametras, tai į metodą persiduoda nuoroda į jį:

```
void metodas(int[] a) {
        a[0] = 100;
}

void kitas() {
        int[] a = {200, 400, 600};
        this.metodas(a);
        System.out.print(a[0]);
}
```

Kokią a[0] reikšmę atspausdins?

Objekto sunaikinimas (object destroy)

Java kalboje nėra galimybės tiesiogiai sunaikinti sukurtą objektą.

Objektas sunaikinamas automatiškai, kai tik nelieka kintamųjų, kurie turi nuorodą į objektą, pvz.:

```
int[] a = {1, 10, 100};
int[] b = a;
a = null;
b = null;
```

Objekto laukų pradinės reikšmės (default values)

- Objekto sukūrimo metu kiekvienas aprašytas laukas įgyja tam tikras reikšmes pagal nutylėjimą (jei nenurodytos kitos reikšmės).
- Toks reikšmės pagal nutylėjimą yra:
- 0 - visiems skaitmeniniams laukams
- false - boolean tipo laukams
- null - tekstinėms eilutėms ir visiems objektams
- ‘\u0000’ - char tipo laukams

Klasės laukai (class fields)

Kuriant objektą, kiekvieną kartą sukuriami ir objekto klasėje aprašyti laukai. T.y. kiekvienas sukurtas objekto egzempliorius gauna tik jam skirtas laukų kopijas.

- Bet galima aprašyti ir tokį lauką, kurio visą laiką bus tik viena kopija nepriklausomai nuo to, kiek klasės egzempliorių yra sukurta. Arba gal dar iš vis nesukurta nei vieno egzemplioriaus.
- Tokį lauką reikia aprašyti su modifikatoriumi **static**.
- Tokius laukus mes vadiname klasės laukais (class variables) Ir jie pasiekiami tiesiai iš klasės:

```
class Zmogus {
        static int kiekis = 0;
}
System.out.println("Viso žmonių " + kiekis);
```

Klasės metodai (class methods)

Jei klasės laukai yra bendri visiems objektams ir yra prieinami iš klasės, tai lygiai tas pats tinka ir klasės metodams, t.y. metodams pažymėtiems modifikatoriumi static. Paprastai tokie metodai būna skirti prieiti prie privačių klasės laukų:

```
private static int kiekis = 0;
public static int kiek() {
        return kiekis;
}
```

Pagalbinės klasės (helper classes)

Klasės metodai taip pat naudojami aprašyti taip vadinamas klases -helperius ar funkcijų konteinerius, pvz:

```
public class Area {
public static double circle(double r) {
        return Math.PI * r * r;
}

public static double triangle(double a, double b, double c) {
        double s = (a + b + c)/2;
        return Math.sqrt(s * (s - a) * (s - b) * (s - c));
}
}
```

Klasė be instance (Class with impossible instance)

Kaip apsisaugoti, kad negalėtumėm sukurti tokios klasės egzemplioriaus?

```
public class Area {
        private Area() {
        }

public static double circle(double r) {
                return Math.PI * r * r;
}

public static double triangle(double a, double b, double c) {
                double s = (a + b + c)/2;
                return Math.sqrt(s * (s - a) * (s - b) * (s - c));
}
}
```

Abstrakti klasė (abstract class)

Abstrakti klasė tai tokia klasė:

- Jos kai kurie metodai nėra iki galo aprašyti, t.y. apie juos žinome tik jų pasiekiamumo tipą, pavadinimą, parametrus ir kokią reikšmę jie grąžina.
- Mes nenorime (!!!), kad galima būtų sukurti tokios klasės objektą ir norime (!!!), kad ji būtų naudojama tik kaip tėvinė klasė kitoms klasėms aprašyti.
- Kad klasė yra abstrakti nurodoma su modifikatoriumi **abstract**

Abstrakti klasė (abstract class)

Abstrakti klasė gali turėti ne iki galo aprašytus metodus, t.y. tokius metodus kuriems nurodyta tik jų pasiekiamumo tipas, pavadinimas, parametrai ir kokią reikšmę jie grąžina:

```
public abstract class Figure {
    private String color;

    public Figure(String color) {
    this.color = color;
    }

    public abstract double getArea();
}
```

Polimorfizmas (polymorphism)

Polimorfizmas – objektiniame programavime naudojama sąvoka, kai operacija (metodas) gali būti vykdomas skirtingai, priklausomai nuo konkrečios klasės (ar duomenų tipo) realizacijos, metodo kvietėjui nieko nežinant apie tokius skirtumus ir galvojant, kad jis operuota tik su bazinės (tėvinės) klasės objektais.

Tai pasiekiama aprašant metodus (operacijas) bazinėje klasėje ir perrašant atitinkamus metodus paveldinčiose klasėse. Metodai, kuriuos galima (paliekant tą patį pavadinimą) perrašyti paveldinčioje klasėje, vadinami virtualiais.

Kai dukterinėje klasėje perrašomas tėvinės klasės metodas, tai galima tą metodą dukterinėje klasėje pažymėti specialia žyme (anotacija) **@Override**

Taip mes pažymime, kad žinome, jog šio metodo aprašas (vardas, grąžinamos reikšmės tipas, parametrai) sutampa su tėvinės klasės metodu.

Tyčia ar netyčia pakeitus tėvinio ar dukterinio metodo aprašą ir atsiradus neatitikimams, mums bus rodomas klaidos pranešimas.

Subtipas/supertipas (subType/superType)

class Mokinys extends Zmogus { … }

Mes žinom, kad Mokinys tai **dukterinė** klasė klasės Zmogus atžvilgiu.
Savo ruožtu Zmogus yra **tėvinė** klasė klasei Mokinys.
Ji taip pat vadinama **bazine (base)** klase klasei Mokinys.
Klasė Mokinys yra **subtipas (subtype)** klasei Zmogus.
Klasė Zmogus yra **supertipas (supertype)** klasei Mokinys.

Abstrakcija (abstraction)

Kiekvienas subtipo kintamasis tuo pačiu yra supertipo. T.y. ten kur galima naudoti supertipo klases kintamuosius ar parametrus, ten taip pat galima naudoti ir subtipo.

Aprašant kintamuosius ar metodų parametrus reikia stengtis naudoti kuo bendresnes klases:

class Circle extends Figure { … }

Ten kur galima naudoti Figure objektus galima naudoti ir Circle objektus, nes kiekvienas Circle objektas yra ir Figure tipo objektas.

Uždaviniai

1. Suraskite kiekvienos klasės visų mokinių pažymių vidurkį ir atspausdinkite kartu su klasės numeriu.
2. Atspausdinkite klases su jų vidurkiais vidurkio mažėjimo tvarka.
3. Sukurkite abstrakčią klasę Figura kurioje būtų aprašyti abstraktūs metodai plotui ir perimetrui paskaičiuoti. Tada sukurkite dukterines klases Apskritimas, Kvadratas, Trikampis(lygiakraštis).
   - Paskaičiuokite kokie turėtų būti visų perimetrai, kad plotai būtų vienodi, tarkime lygūs 100.
   - Paskaičiuokite kokie turėtų būti visų plotai, kad perimetrai būtų vienodi, tarkime lygūs 100.

   Pastaba: apsirašykite Figura klasėje tokius abstrakčius metodus ir po to juos aprašykite dukterinėse klasėse, kad kaip parametrą pateikus plotą arba perimetrą, jie paskaičiuotų ir nustatytų atitinkamos figūros kraštinę ar spindulį.