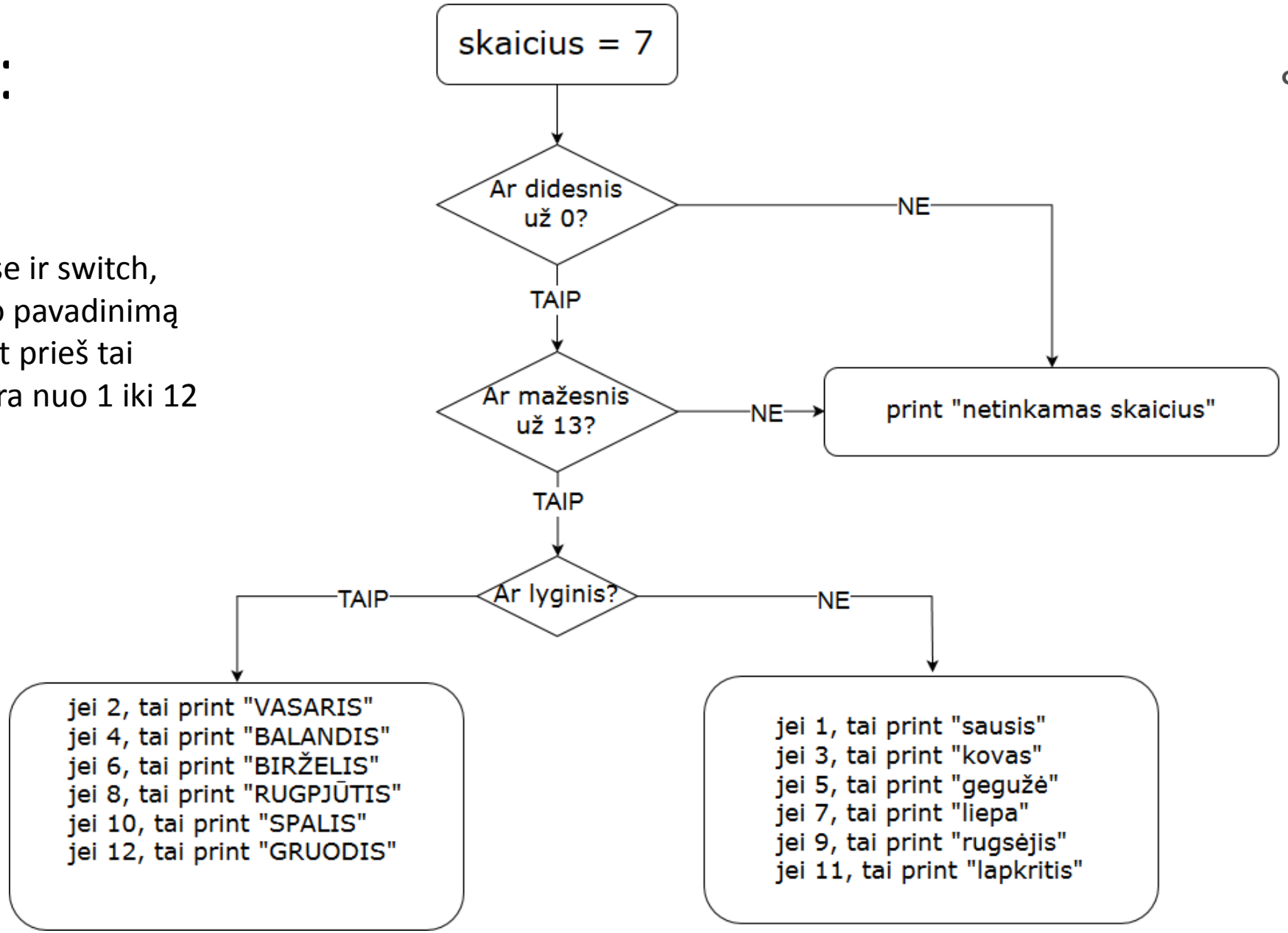
Užduotis 1: if..else

Parašyti programą su if-else ir switch, kuri atspausdintų mėnesio pavadinimą pagal nurodytą skaičių, bet prieš tai patikrintų ar tas skaičius yra nuo 1 iki 12



Užduotis 2: eilutės formatavimas

- Vartotojas įveda 10 vardų
- Tuos vardus sudeda į String tipo masyvą
- Atspausdina visus masyvo elementus tokiu formatu:
  - {vardas vien didžiosiomis raidėmis}-{vardo simbolių ilgis}-{elemento masyve numeris}
  - Pvz: "JURGIS-6-0"

Užduotis 3: for ciklai

Naudojant ciklą FOR atspausdinti tokią figūrą:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

Užduotis 4: for/if..else

- Programa naudotojui leidžia įvesti 10 skaičių
- Įvedus skaičius programa turi atspausdinti didžiausią ir mažiausią skaičių iš įvestų

Užduotis 5: while ciklai

- Vartotojas turi vesti skaičius tol, kol įves neigiamą
  - Programa paprašo įvesti skaičių
  - Jei įvestas skaičius yra teigiamas programa vėl paprašo įvesti skaičių
  - ...
  - Įvedus neigiamą skaičių programa sustoja.

Užduotis 6: klasės ir metodai

- Klasė turi **main** metodą, kuris kviečia jūsų sukurtus du static metodus:
  - **m1()** ir po to **m2()**
  - metodai turi gražinti **double** rezultatą
  - **main** metode: metodų rezultatai priskiriami kintamiesiams **d1**, **d2**
  - **main** metodas išspausdina jų sumą
- **m1()** - metodas gražina **double** tipo reikšmę 123.5;
- **m2()** - metodas gražina **double** tipo reikšmę kurią įvedame į programą naudodami:

```
Scanner scanner = new Scanner(System.in);
System.out.println("Įveskite skaičių:");
double a = scanner.nextDouble();
```

Užduotis 7: klasės ir metodai

- Parašyti programą su klase ***Darbuotojas*** kuri turi tokius metodus:
  - ***gautiInfo()*** , kuris atspausdina atlyginimą ir dirbtų valandų skaičių
  - ***pridėtiAtlyginimą()*** , kuris prideda 10 Eur, jei darbuotojo alga neviršija 500 Eur
  - ***pridėtiDarbą()*** , kuris prideda 5 Eur prie algos ir prideda per metodo parametrą gautą valandų skaičių
- Klasė turi du kintamuosius: ***atlyginimas*** ir ***darbas***.
- Klasė ***Darbuotojas*** turi turėti bent 2 konstruktorius.
- Darbuotojas tipo objektą susikurti kitoje klasėje ir paeksperimentuoti su ***package-private***, **private**, **protected** ir **public** modifikatoriais perkeliant klasę ***Darbuotojas*** į kitą paketą.
- Klasę ***Darbuotojas*** testuoti kitoje klasėje, kuri turi ***main*** metodą.

Užduotis 8: statiniai ir nestatiniai metodai

- Reikalingos dvi klasės
- Vienoje iš klasių:
  - Sukurkime ne-**static int** tipo kintamajį
  - Sukurkime ne-**static** metodą **setValue(int i)** su parametru **int i** kuris išspausdina **i** reikšmę ir priskiria ją objekto kintamajam
  - Sukurkime **static** metodą **staticMethod(int j)** su parametru **int j**, kuris išspausdina **j** reikšmę ir pabando priskiria ją objekto kintamajam
- Kitoje klasėje:
  - Sukuriamas **main** metodas, kuris pabandys iškviesti pirmos klasės: **staticMethod(int j)** su reikšme 10 ir metodą **setValue(int i)** su reikšme 10

Užduotis 9: statiniai metodai ir kintamieji

- Reikalingos dvi klasės
- Vienoje iš klasių:
  - Sukurkime **static int** kintamajį
  - Sukurkime konstruktorių, kuris išspausdina statinio kintamojo reikšmę ir padidina vienetu
  - Sukurkime static metodą **isvalyti()**, kuris išspausdina statinio kintamojo reikšmę ir priskiria jam **0**
- Kitoje klasėje:
  - Sukuriamas main metodas, kuris:
    - Sukuria 5 pirmos klasės objektus/egzempliorius
    - Išspausdina pirmos klasės statinio kintamojo reikšmę
    - Iškviečia statini metodą **isvalyti()**
    - Ir vėl išspausdina pirmos klasės statinio kintamojo reikšmę

Užduotis 10: final

- Sukurti kalse **TestFinal** su dviem kintamaisiais. Klasė taip pat turi turėti metodą **keisti**, kuriame kintamiesiams priskiriamos perduotos reikšmės ir išspausdina objekto kintamųjų reikšmes.
  - Main metode sukurti keletą **TestFinal** objektų ir jiems iškviesti metodą **keisti**
  - Paskelbti vieną iš kintamųjų **final**

Užduotis 11: static final

- Sukurti klasę **StaticTestFinal** su dviem **static** kintamaisiais. Klasė taip pat turi turėti metodą **priskirk** kuriame kintamiesiams priskiriamos perduotos reikšmės ir išspausdinamos objekto kintamųjų reikšmės.
  - **main** mentode sukurtt keletą **StaticTestFinal** objektų ir jiems iškviesti metodą **priskirk**
  - Paskelbti vieną iš kintamųjų **final**.

Užduotis 12: metodai

- Metodas/Funkcija apskaičiuojanti apskritimo perimetrą (parametras `float spindulys`)

$$2\pi r$$

Užduotis 13: klasės

- Sukurti klasę **StudentData**. Klasė turi turėti laukus: **studento id**, **vardas** ir **metai**.
- Laukai turi būti nepasiekiami kitoms klasėms.
- Laukai: studento identifikatorius ir metai - turėtu būti apsaugoti nuo keitimo/modifikavimo (uždrausta keisti jų reikšmes)
- Klasė turi turėti:
  - metodus kurie grąžina laukų reikšmes “getters”
  - metodą kuris išspausdina klasėje saugomą informaciją
  - galimybę pakeisti vardą
- ***main*** metode sukurti keleta objektų/egzempliorių. Pabandyti pasiekti laukus, pakeisti jų reikšmes, pakeisti vardą.

Užduotis 14: data ir laikas

- Paprašyti vartotojo įvesti laiką
- Prie įvesto laiko pridėti 2 val. ir 15 min.
- Gautą laiką atspausdinti į ekraną
- Patikrinti, ar įvestas laikas yra ankstesnis už dabarties laiką

Užduotis 15: eilutės formatavimas

- Naudojant String.format atspausdinti tokį tekstą:

```
|                  Aš|
|               tikrai|
|            išmoksiu|
|         programuoti|
|Java                 |
```

- Eilutės ilgis 20
  - String.format(…., …, …)
  - %20s
  - \n

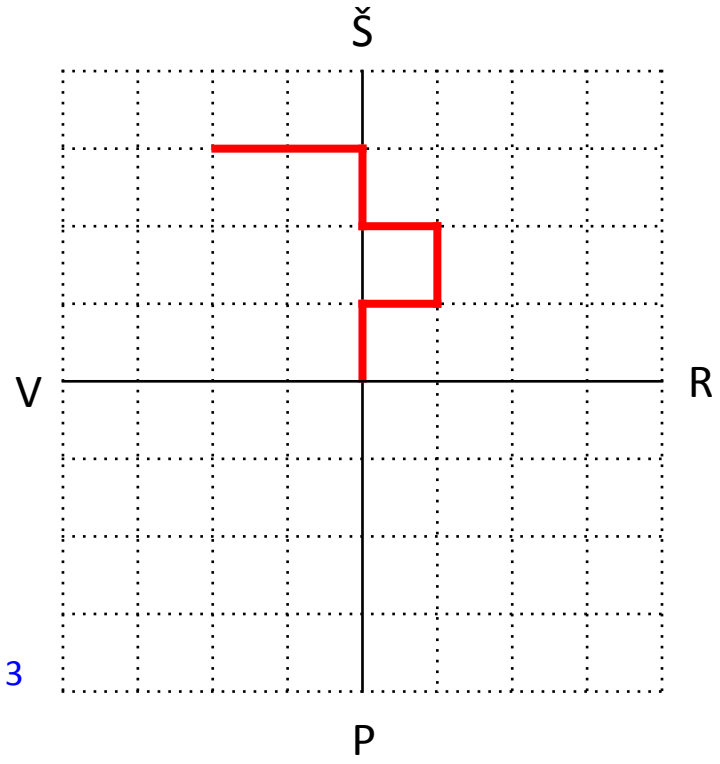
Užduotis 16: Tankas

- Klasė: **Tankas**
- Metodai: **pirmyn**, **atgal**, **kairėn**, **dešinėn**, **šūvis**, **info**, ...
- Klasės kintamieji:
  - saugoti koordinates,
  - saugoti kryptį,
  - saugoti šūvių skaičių į kiekvieną kryptį.
- Tankas gali judėti pirmyn (į Šiaurę), dešinėn (į Rytus), atgal (į Pietus), kairėn (į Vakarus) per vieną poziciją. Pvz. „tankas pajuda kairėn“, tai reiškia jis pasisuko 90 laipsnių ir pajudėjo per vieną vienetą į Vakarus.
- Tankas gali šaudyti TIK ta kryptimi, į kurią jis yra pasisukęs.
- Metodas **info()** turi parodyti:
  - į kurią kryptį tankas šiuo metu yra pasisukęs,
  - kokios yra jo koordinatės,
  - kiek iš viso atliko šūvių,
  - kiek atliko šūvių į kiekvieną kryptį atskirai.

Tanko judėjimo pavyzdys

```
Tankas tankas = new Tankas();
tankas.pirmyn();
tankas.desinen();
tankas.pirmyn();
tankas.suvis();
tankas.suvis();
tankas.kairen();
tankas.suvis();
tankas.pirmyn();
tankas.info();
tankas.kairen();
tankas.kairen();
tankas.suvis();
tankas.info();
```

```
Tankas pajuda į Šiaurę (0;1)
Tankas pajuda į Rytus (1;1)
Tankas pajuda į Šiaurę (1;2)
Šūvis į Šiaurę
Šūvis į Šiaurę
Tankas pajuda į Vakarus (0;2)
Šūvis į Vakarus
Tankas pajuda į Šiaurę (0;3)
INFO: Tanko koordinatė: (0;3), kryptis: SIAURE
INFO: Tanko šūviai: 2 į Šiaurę, 0 į Rytus, 0 į Pietus, 1 į Vakarus.  Iš viso šūvių: 3
Tankas pajuda į Vakarus (-1;3)
Tankas pajuda į Vakarus (-2;3)
Šūvis į Vakarus
INFO: Tanko koordinatė: (-2;3), kryptis: VAKARAI
INFO: Tanko šūviai: 2 į Šiaurę, 0 į Rytus, 0 į Pietus, 2 į Vakarus.  Iš viso šūvių: 4
```



Interaktyvi programa

- Programa išspausdina galimas komandas [v] – tankas judės į Vakarus, [x] – baigti programą, ...
- Programa priima vartotojo įvestą komandą.
- Programa paskaičiuoja dabarties laiką.
- Pagal vartotojo įvestą komandą programa atlieka veiksmą.
- Kartu su pranešimu programa atspausdina laiką (arba datą ir laiką), kada veiksmas buvo atliktas, pvz.:

  [22:30] Tankas pajuda į Šiaurę (0;1)

  [22:32] Tankas pajuda į Rytus (1;1)
- Atlikus veiksmą programa toliau laukia kitų vartotojo veiksmų.

Interaktyvi programa

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

	Tankas tankas = new Tankas();
	Scanner sc = new Scanner(System.in);
	boolean runProgram = true;
	while(runProgram) {
		// atspausdinamas komandų sąrašas
		// gaunamas/paskaičiuojamas dabarties laikas
		// nusprendžiama, kokią komandą įvykdyti
		// įvykdoma komanda
	}
	sc.close();

}
```

Pasirinkite:

[s] - ejimas i Siaure

[r] - ejimas i Rytus

[p] - ejimas i Pietus

[v] - ejimas i Vakarus

[*] - suvis

[i] - info

[x] - pabaiga

s

Tankas pajuda į Šiaurę (0;1)

Pasirinkite:

[s] - ejimas i Siaure

[r] - ejimas i Rytus

[p] - ejimas i Pietus

[v] - ejimas i Vakarus

[*] - suvis

[i] - info

[x] - pabaiga

r

Tankas pajuda į Rytus (1;1)

Pasirinkite:

[s] - ejimas i Siaure

[r] - ejimas i Rytus

[p] - ejimas i Pietus

[v] - ejimas i Vakarus

[*] - suvis

[i] - info

[x] - pabaiga

*

Šūvis į Rytus

Pasirinkite:

[s] - ejimas i Siaure

[r] - ejimas i Rytus

[p] - ejimas i Pietus

[v] - ejimas i Vakarus

[*] - suvis

[i] - info

[x] - pabaiga

s

Tankas pajuda į Šiaurę (1;2)

Pasirinkite:

[s] - ejimas i Siaure

[r] - ejimas i Rytus

[p] - ejimas i Pietus

[v] - ejimas i Vakarus

[*] - suvis

[i] - info

[x] - pabaiga

i

INFO: Tanko koordinatė: (1;2), kryptis: SIAURE

INFO: Tanko šūviai: 0 į Šiaurę, 1 į Rytus, 0 į Pietus, 0 į Vakarus. Iš viso šūvių: 1

Pasirinkite:

[s] - ejimas i Siaure

[r] - ejimas i Rytus

[p] - ejimas i Pietus

[v] - ejimas i Vakarus

[*] - suvis

[i] - info

[x] - pabaiga

x

Užduotis 17: Koordinatės

- Sukurti klasę **Koordinate**, kuri turi **private** kintamuosius **x** ir **y** bei konstruktorių priimanti du parametrus ir **get**terius/**set**terius
- Sukurti sąrašą, kuris saugo **Koordinate** tipo elementus
- Į sąrašą įdėti kelis elementus (pvz **sarasas.add(new Koordinate(2, 5));**)
- Kordinatės: (1; 5), (5; 9), (4; 0), (0; 0), (9; 1), …
- Atspausdinti koordinates
- Rasti kelinta sąraše yra (0; 0) koordinatė
- Pakeisti šio (0; 0) objekto koordinates į (1; 1)
- Atspausdinti koordinates

Užduotis 18: Map

- Sukurti HashMap<Integer, String> ir įdėti į jį key-value porų **.put(..., ...)**
  - 11 -> ”vienuolika”, 12 -> “dvylika”, 100 -> “šimtas”, ...
- Atspausdinti HashMap
- Patikrinti, ar HashMap‘e yra pora pagal raktą **.containsKey(...)** ir pagal reikšmę **.containsValue(...)**
- Gauti reikšmę pagal raktą ir ją atspausdinti
- Atspausdinti visas HashMap‘o reikšmes tokiu formatu:
  - Raktas: 11, Reikšmė: Vienuolika
  - Raktas: 100, Reikšmė: Šimtas
  - ...

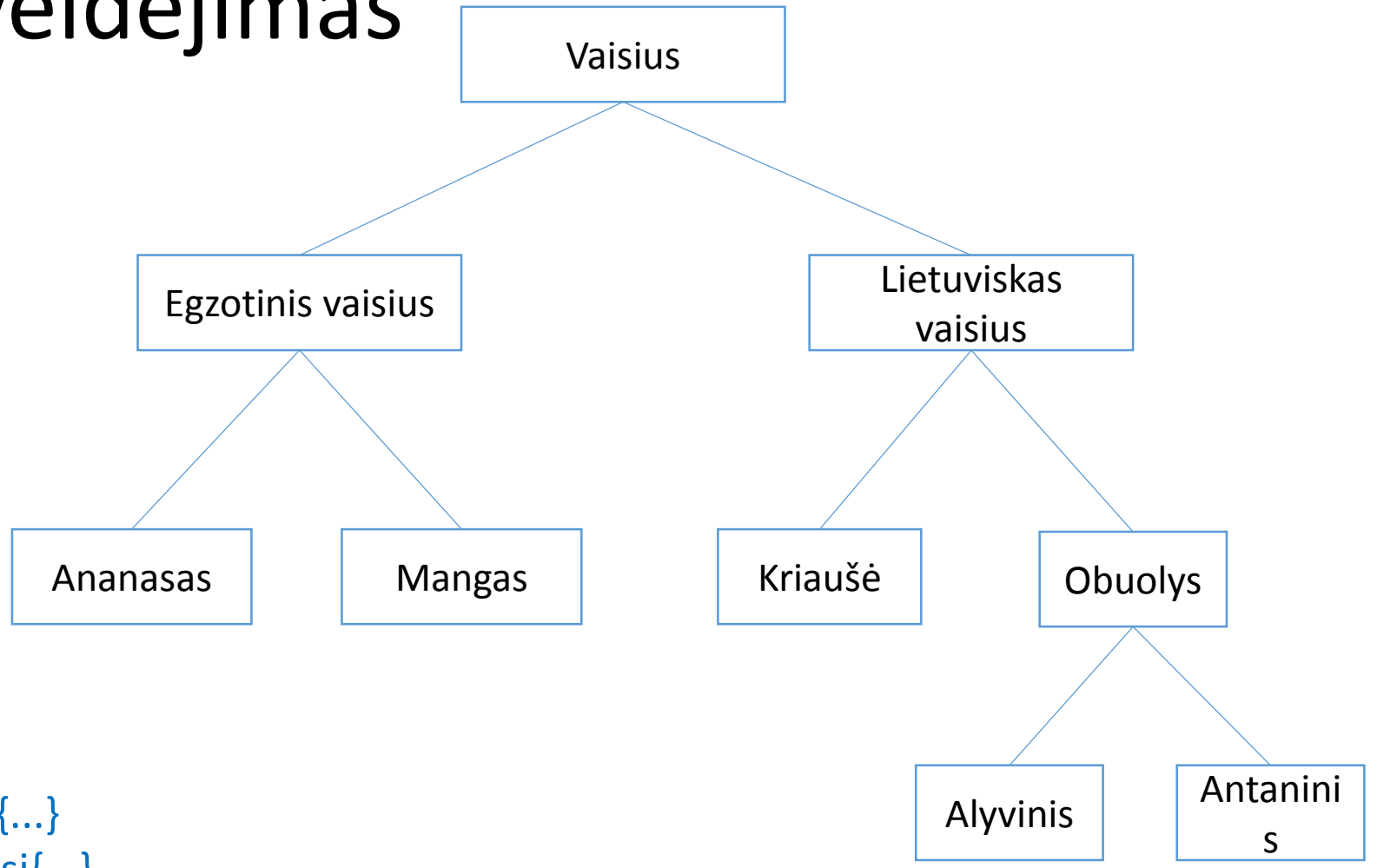
Užduotis 19: Map

- Toliau tęsti pirmoje užduotyje naudotą programą
- Padaryti metodą **spausdintiMap**, kuris priima paduotą HashMap ir jį atspausdina ankstesnėje užduotyje aprašytu formatu
  void spausdintiMap(HashMap<Integer, String> mapas) {...}
- Metodą iškviesti paduodant jam užpildytą HashMap
- Iš HashMap‘o pašalinti porą pagal raktą, pvz. 11.
- Atspausdinti HashMap tuo pačiu formatu
- Išvalyti visą HashMap
- Atspausdinti HashMap tuo pačiu formatu

Užduotis 20: Map

- Sukurti HashMap<Integer, String> ir įdėti į jį key-value porų **.put(..., ...)**
  - 11 -> ”vienuolika”, 12 -> “dvylika”, 100 -> “šimtas”, ...
- Iš esamo HashMap‘o padaryti kitą HashMap<String, Integer>, kuris būtų:
  - „vienuolika“ -> 11, “dvylika” ->12, …
- Papildyti programą nauju metodu, kuris mokėtų atspausdinti tokį HashMap‘ą pagal ankstesnį formatą
- Vėliau atspausdinti tik HasMap‘o raktus

Užduotis 21: Paveldėjimas



Padaryti tokią klasių hierarchiją

class AlyvinisObuolys extends Obuolys {...}
class Obuolys extends LietuviskasVaisiusi{...}
...

Užduotis 22: Paveldėjimas

- Pirmos užduoties tęsinys
- Kiekviena klasė turi metodą ***kasAsEsu()***, kuris atspausdina „aš esu obuolys“, „aš esu egzotinis vaisius“ ir t.t. Naudoti anotaciją @Override
- Sukurti kiekvienos klasės objektą ir kiekvienam objektui iškviesti šį metodą, pvz.:

```
Kriause k1 = new Kriause();
k1.kasAsEsu();
```

- Iš klasės Obuolys pašalinti metodą *kasAsEsu()* ir pažiūrėti, ką atspausdins *obuolys.kasAsEsu()*

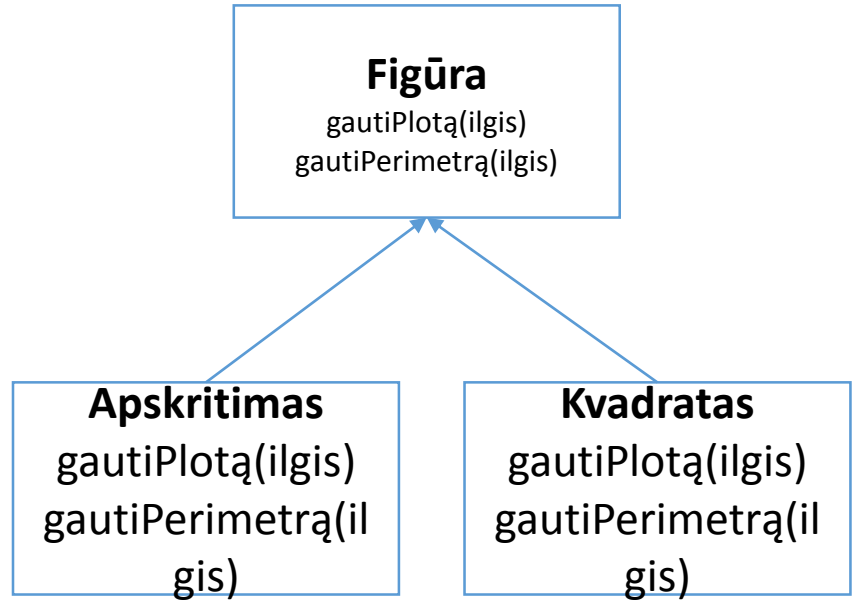
Užduotis 23: toString

- Sukurti klasę **Asmuo** su klasės kintamaisiais **vardas** ir **pavardė** (konstruktorius, get, set metodai)
- Užkloti **Object** klasės metodą **toString()**, kuris grąžintų informaciją apie asmenį skaitomu formatu, pvz.
  - „Asmuo [vardas=Antanas, pavardė=Antanaitis]”
- Klasėje su **main** metodu sukurti Asmuo tipo objektą arba tokių objektų sąrašą. Ir tokį objektą/sąrašą atspausdinti.

Užduotis 24: Paveldėjimas

- **Figūra** yra tėvinė klasė su dviem abstrakčiais metodais **gautiPlotą** ir **gautiPerimetrą**
- Klasės **Apskritimas** ir **Kvadratas** paveldi klasę **Figūra** ir įgyvendina metodus **gautiPlotą** ir **gautiPerimetrą**
- **Main** metode sukurti du objektus ir iškviesti jiems ploto ir perimetro gavimo metodus

```
Figūra a = new Apskritimas();
Figūra k = new Kvadratas();
```

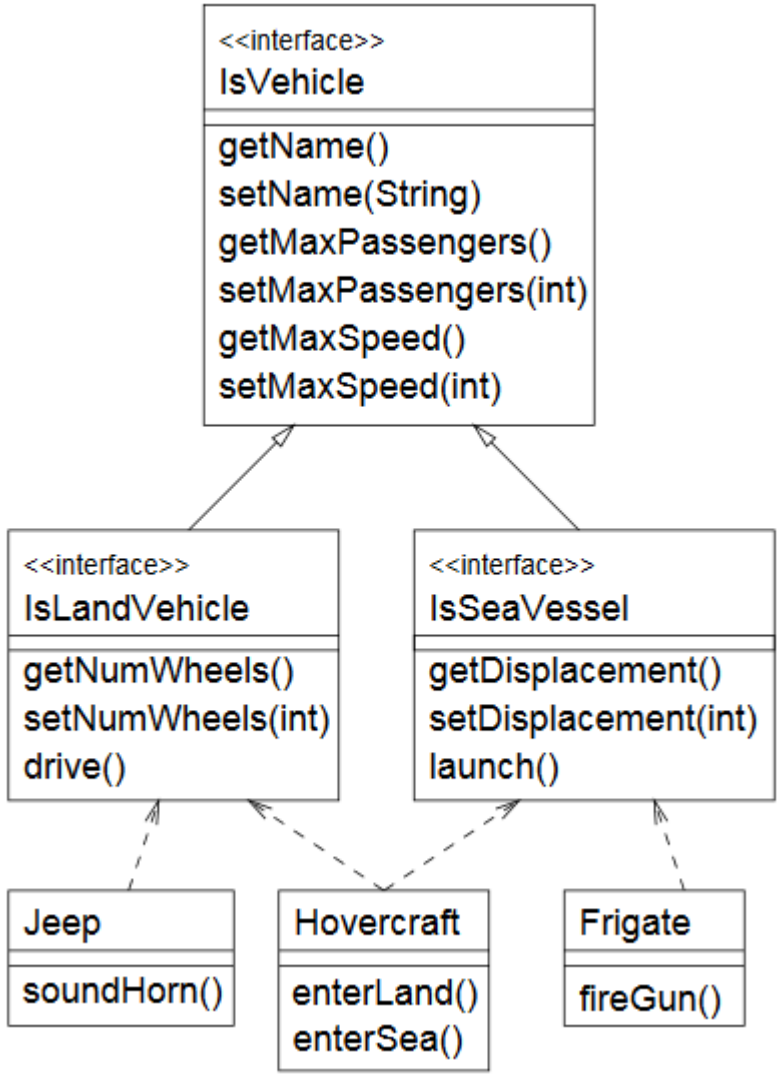


$$A = \pi r^2$$
$$C = 2\pi r$$

$$A = a^2$$
$$P = 4a$$

Užduotis 25: Interfeisai

- Sukurti tokią klasių ir interfeisų hierarchiją
- Pvz. metodas *drive()* tegul atspausdina pranešimą, kad transporto primonė važiuoja
- **Main** metode sukurti keletą įvairų objektų ir sudėti juos į masyvą arba ArrayListą
- Iteruoti visus masyvo objektus ir patikrinti kokio tipo tas objektas yra (instanceof)
- Kiekvienam elemetui kviesti jam galimus metodus prieš tai pakeitus jo tipą (cast)



Užduotis 26: Interfeisai

- Sukurti Interfeisą **IsEmergency**, kuris nepaveldi jokio kito interfeiso ir turi metodą **soundSiren**
- Sukurti klasę **PoliceCar**, kuri implementuoja šiuos interfeisus: **IsEmergency** ir **IsLandVehicle**
- Pridėti keletą **PoliceCar** objektų į masyvą

Užduotis 27: kompozicija ir paveldėjimas

- Sukurti klasę **X**
- Klasė **X** turi turėti metodą **metodasX()**, kuris atspausdina „Iškviestas metodas X“
- Sukurti dvi klases **A** ir **B**
- Kompozicija:
  - **A** klasė turi kintamąjį **X**, taip pat **getX** ir **setX** metodus
- Paveldėjimas:
  - **B** klasė paveldi klasę **X**
- Atskiroje klasėje **Main** metode sukurti **A** ir **B** objektus ir abiem atvejais iškviesti **metodasX()**

Užduotis 28: Konstruktoriai

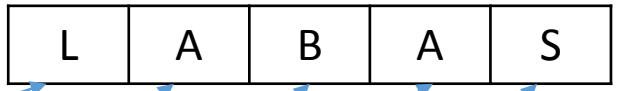
- Klasės pavadinimas yra viena raidė
- Klasės konstruktorius atspausdina klasės pavadinimą, t.y. raidę, pvz:

```
class L {
	L() {
		System.out.print("L");
	}
}
```

- Turint tokią sąlygą ir turint panaudoti bent du paveldėjimus atspausdinti žodį:
  - **LABAS**
- **Main** metode bus kviečiama:
  - **new L();**

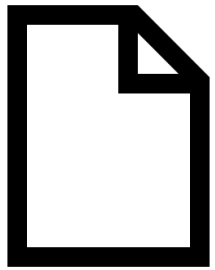
Užduotis 1

```
class L {
	L() {
		System.out.print("L");
		new B();
		new S();
	}
}

class A {
	A() {
		System.out.print("A");
	}
}

class B extends A{
	B() {
		System.out.print("B");
	}
}

class S extends A {
	S() {
		System.out.print("S");
	}
}
```



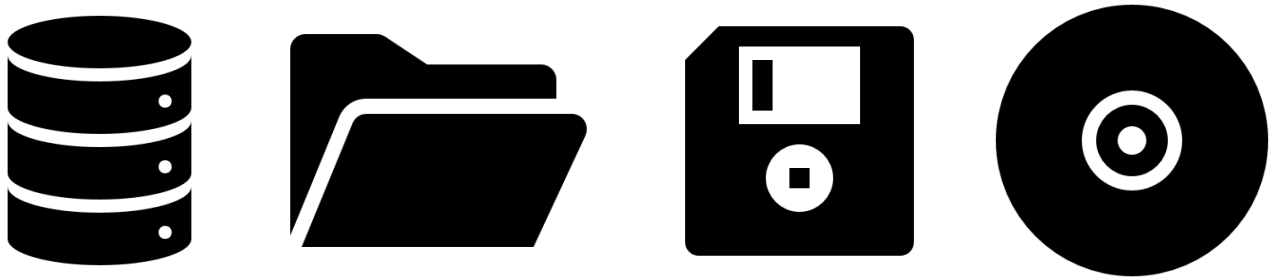
Užduotis 29 [1/6]

- Sukurti klasę **Info**
  - Klases kintamieji (private):
    - **id** - skaičius
    - **tekstas** – eilutės tipas
  - Konstruktorius su dviem parametrais
  - Get ir Set metodai
  - toString
  - Metodas **arYraTekste**, kuris priima eilutes tipo parametrą žodis, patikrina ar paduotas žodis yra tekste ir grąžina **true/false**

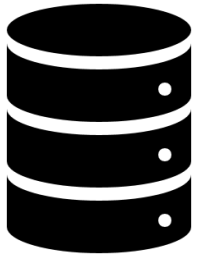


Užduotis 29 [2/6]

- Sukurti interfeisą **Saugykla** su tokiais metodais:
  - **saugotiInfo** – priima **Info** tipo parametrą ir nieko negrąžina
  - **rastiInfo** – priima **id** (skaičius) ir grąžina **Info** tipo objektą
  - **rastiInfo** – priima **zodis** (eilutės tipas) ir grąžina **Info** tipo objektą

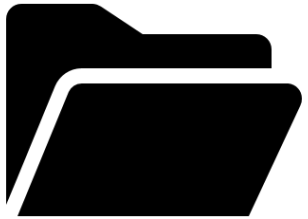


Užduotis 29 [3/6]



- Sukurti klasę **DuomenuBaze**, kuri įgyvendina **Saugykla** interfeisą
- Klasė turi turėti **HashMap**, skirtą **Info** objektų saugojimui
- Raktas yra **id**, o reikšmė – **Info** objektas
- **saugotiInfo** metodas išsaugo gautą **Info** objektą į **HashMap**
- Pirmasis **rastiInfo** metodas gauna **Info** iš map‘o pagal key (**id**) ir grąžina jį
- Antrasis **rastiInfo** metodas gauna **Info** iš map‘o iteruodamas kiekvieną reikšmę ir tikrindamas ar per parametrus gautas žodis yra **Info** objekto tekste. Grąžina pirmą rastą **Info** objektą
- Visi metodai atspausdina kas buvo padaryta: „Issaugota i duomenu baze“, „Rasta duomenu bazeje pagal id“, „Rasta duomenu bazeje pagal zodi“

Užduotis 29 [4/6]



- Sukurti klasę **NutolesDiskas**, kuri įgyvendina **Saugykla** interfeisą
- Klasė turi turėti **ArrayList**, skirtą **Info** objektų saugojimui
- **saugotiInfo** metodas išsaugo gautą **Info** objektą į **ArrayList**
- Pirmasis **rastiInfo** metodas gauna **Info** iš sąrašo iteruodamas kiekvieną elementą ir tikrindamas ar per parametrus gautas **id** yra lygus **Info** objekto **id**. Grąžina pirmą rastą **Info** objektą
- Antrasis **rastiInfo** metodas gauna **Info** iš sąrašo iteruodamas kiekvieną elementą ir tikrindamas ar per parametrus gautas žodis nėra **Info** objekto tekste. Grąžina pirmą rastą **Info** objektą
- Visi metodai atspausdina kas buvo padaryta: „Issaugota i nutolusi diska“, „Rasta nutolusiame diske pagal id“, „Rasta nutolusiame diske pagal zodi“

Užduotis 29 [5/6]

- Sukurti klasę **Programa** su metodu **main**
- Klasėje taip pat turi būti dar trys statiniai metodai
- Metodas **saugoti**
  - Nieko negrąžina, bet priima **Saugykla** ir **Info** parametrus
  - Metodas iškviečia Saugyklos metodą **saugotiInfo**
- Metodas **rastiPagalId**
  - Nieko negrąžina, bet priima **Saugykla** ir **id** parametrus
  - Metodas iškviečia Saugyklos metodą **rastiInfo**
  - Atspausdina rastą Info objektą
- Metodas **rastiPagalZodi**
  - Nieko negrąžina, bet priima **Saugykla** ir **zodis** parametrus
  - Metodas iškviečia Saugyklos metodą **rastiInfo**
  - Atspausdina rastą Info objektą

Užduotis 29 [6/6]

- **Main** metode sukurti dviejų tipų saugyklas – nutolusį diską ir duomenų bazę
- Sukurti keletą **Info** tipo objektų
- Iškviesti **saugoti**, **rastiPagalid** ir **rastiPagalZodi** metodus

Užduotis 30: null

- Sukurti klasę ***Miestas***, kuri turėtų ***String*** tipo klasės kintamąjį ***pavadinimas***.
- Sukurti klasę ***Adresas***, kuri turėtų ***Miestas*** tipo klasės kintamąjį ***miestas***.
- Sukurti klasę su ***main*** metodu.
- Šalia ***main*** metodo sukurti kitą statinį metodą pavadinimu ***gautiMiestoPavadinima***, kuris priima ***Adresas*** tipo objektą ir grąžina ***String*** tipo rezultatą. Metodas turi išgauti miesto pavadinimą iš adreso. Metode turi būti apsisaugota nuo neegzistuojančių objektų (***null***).
- Main metode sukurti ***Adresas*** tipo objektą su miestu bei pavadinimu, iškviesti metodą ***gautiMiestoPavadinima*** ir atspausdinti rezultatą
- ***Nepamiršti***: klasės – public, klasės kintamieji – private, klasės konstruktoriai – public, konstruktoriai gali priimti parametrus, public get ir set metodai turi būti visiems klasės kintamiesiems

Užduotis 32

- Sukurti klasę ***Lektuvas*** su metodu ***isskleistiVaziuokle***, kuris nereikalauja parametrų ir nieko negrąžina
- Metodas ***isskleistiVaziuokle*** sugeneruoja atsitiktinį skaičių iš intervalo **[0; 10]**
  - ***rand.nextInt(10);***
- Sugeneruotas skaičius tame pačiame metode naudojamas dalybai – bet koks skaičius dalijamas iš sugeneruoto skaičiaus. Po dalybos atspausdinamas pranešimas „**OK: važiuoklė sėkmingai išskleista**“
- Kitoje klasėje ***main*** metode sukurti ***Lektuvo*** objektą ir penkis kartus iškviesti metodą ***isskleistiVaziuokle***

Užduotis 33

- Ankstesnėje užduotyje parašytą programą apsaugoti nuo visiško programos „sulūžimo“
- Programa gali „sulūžti“ dalybos iš nulio atveju, tačiau turėtų sugebėti toliau vykdyti kitas užduotis (metodo ***isskleistiVaziuokle*** iškvietimus)
- Sulūžus programai reikia atspausdinti „**ERROR: važiuoklės išskleisti nepavyko**“
- Panaudoti ***try-catch*** bloką

Užduotis 34

- Į ***try*** bloką įdėti tokias eilutes:

```
if (x == 5) {
    "skrendu".charAt(20);
}
```

- Paleisti programą
  - Apsaugoti programą nuo naujo tipo klaidos
  - Naujos klaidos atveju vartotojui pranešti „**ERROR: nepavyko išskleisti važiuoklės dėl kitos klaidos**“
  - Nepriklausomai klaida įvyko ar ne, atspausdinti pranešimą „**INFO: lektuvas skrenda**“ naudojant ***finally*** bloką

Užduotis 35

- Sukurti išimities klasę ***VaziuoklesIsskleidimoKlaida***, kuri paveldi ***Exception*** ir turi klasės kintamąjį ***priezastis***
- Šalia metodo **isskleistiVaziuokle** sukurti metodą ***vaziuokle***, kuris generuoja atsitiktinį skaičių intervale **[0; 10]** ir priklausomai nuo skaičiaus iššaukia ankstesniame žingsnyje apsirašytą klaidą su priežastimi. Apdoroti (su if arba switch) 0, 1, 2 skaičius „neatsidare durys“, „nenusileido ratas“ ir pan.
- Metode **isskleistiVaziuokle** pakeisti programos kodą pagal žemiau pateiktą ir pritaikyti klaidų „gaudymą“ pranešant, kad važiuoklė neišsiskleidė dėl konkrečios priežasties

```
int i = 4 / x;
if (x == 5) {
  "skrendu".charAt(20);
}
```

→ `vaziuokle();`

Užduotis 36

- Sukurti klases ***LaikinaKlaida*** ir ***SvarbiKlaida***, kurios paveldi klasę ***VaziuoklesIsskleidimoKlaida***
- Metode ***vaziuokle*** kuriant išimtis panaudoti skirtingo svarbumo klaidas
- Metode ***isskleistiVaziuokle*** taip pat prisitaikyti prie klaidų svarbumo pridededant prie esamų pranešimų:
  - **Svarbumas: laikina**
  - **Svarbumas: svarbi**
  - **Svarbumas: nežinoma**

Užduotis 37: varargs

- Parašyti metodą, kuris išspausdina visų perduotų objektų toString() rezultatus
- Pvz. į metodą paduokime (siandienosData, new ManoKlase(), “Labas“)
- Iškviesti metodą keliais skirtingais būdais

Užduotis 38: vidurkis

- Parašyti metodą, kuris priima sveikų skaičių masyvą ir grąžina masyvo skaičių vidurkį
- Pvz. metodui paduodame [2, 34, 65, 70] ir metodas grąžina 85.5