**Klasės kūrimas**

matomumas class KlasėsVardas{....}

***pvz:***

public class TestClass{ ... }

**Metodo kūrimas:**

matomumas grąžinamasTipas metodoVardas (paramTipas vardas, paramTipas vardas2){ ....}

***pvz:***

// void metodas nieko negražina
public void testMethod(int id, String name){...}
public int testMethod(){...}
// private matomas tik klasėje
private int testMethod(Person person){...}
public static void testMetod(){...}
// protected matomas tik klasėje, kuri paveldi tėvinę klasę arba toje pačioje klasėje
protected void testMethod(int[] mas){...}

---

**Objekto kūrimas:**

KlasėsVardas objektoVardas = new KlasėsVardas();

***pvz:***

TestClass myObject = new TestClass();

---

**Metodo iškvietimas:**

***Iš kitos klasės***

//jei nieko negrąžina
objektoVardas.metodoVardas(paramTipas paramVardas);
// jei grąžinama reikšmė
gražinaTipas lokalusVardas = objektoVardas.metodoVardas();

***pvz:***

//metodas void nieko negrąžina
myObject.testMethod(29, "Petras");
int response = myObject.testMethod();
//galime kviesti metodą metodo parametruose. Svarbu turi sutapti
//grąžinamo metodo tipas ir kviečiamo metodo parametro tipas
myObject.testMethod(myObject.testMethod2(„Petras"));
//statiniai metodai negali tiesiogiai iškviesti tos pačios klasės
//nestatinių metodų(jiems reikia kurti klasės okejktą), o nestatiniai
//statinius gali

***Iš tos pačios klasės***

metodoVardas(kintamojoVardas);

***pvz:***

testMethod(„Petras");
testMethod();
int result = testMethod(24, 298);

---

**Konstruktorius:**

// tai metodas, kuris nieko negrąžina, yra tokiu pat vardu, kaip klasė.
//Galima paduoti parametrus. Iškviečiamas, kai kuriamas klasės
//objektas. By default klasė turi tuščią konstruktorių
matomumas klasėsVardas(kintTipas kintVardas){ ... }

***pvz:***

public class TestClass{
    public TestClass(int val, String name){ ... }
    public TestClass(){ ... }
}

**Masyvai:**

// visada privaloma nurodyti masyvo ilgį. Masyvo indeksai yra nuo
//nulio iki masyvo ilgis - 1.
tipas[] masyvoVardas = new tipas[masyvo ilgis];

***pvz:***

int[] intMas = new int[5];
String[] stringMas = new String[10];
//reikšmės paėmimas iš masyvo
int val = intMas[0];
String val = stringMas[1];
// reikšmės įdėjimas į masyvą
intMas[3] = 152;
stringMas[4] = "Sigis";

---

**Sąlygos sakiniai:**

// į if bloką pateksime tik tada, kai sąlyga teisinga.

**if:**

if(5 < 10){ ...}
// į else bloką patenkame, jei if sąlyga neteisinga
if(5 > 10){ ...}else{ ....}

**else if:**

// jei yra kelios if else sąlygos, patenkame į pirmą tinkamą ir kitas
//nebežiūrime
if(10 > 20){...
}else if(10 < 20){...
}elseif(10 != 20){...}

**swich/case:**

// naudojamas su primityviais tipais, kaip int, String. Break nutraukia
// tolimesnius veiksmus, kai rastas atitikmuo
String action = "suma";
swich(action){
  case „dalyba":
      // galimas veiksmų atlikimas ar metodų kvietimas
      break;
  case „suma":
      // galimas veiksmų atlikimas ar metodų kvietimas
      break;
  default:
      // atliekamas tik tada kai nėra atitikimų
      Break;
}

---

**Ciklai:**

***for ciklas:***

for(tipas kintVardas = pradinisKintamasis; kintVardas < kiek suksimės; per kiek paeisim prie kintVardas)
for(Tipas vardas: perKaSuksimes){...}

***pvz:***

for(int i = 0; i < 50; i++) { ... }
for(String name: names){ ... }

***while ciklas:***

while(iteruosime, kol sąlyga bus true){...}

***pvz***

int i = 0;
// sukamės tol, kol i mažiau nei 5 ir pridedame po 1
while(i < 5){ i++;}

**do/while ciklas:**

do{
  // veiksmas atliekamas bent vieną kartą
}while(iteruosime kol bus true);

***pvz:***

int i = 0;
do{
 System.out.println(i++);
}while(i < 5);

---

**Kintamieji:**

tipas kintamojoVardas = reikšmė

***pvz:***

int value = 10;
String value = "Petras";
char value = 'a';
boolean value = false;

---

**Dinaminiai masyvai:**

***List:***

//Listas paprasčiausia kolekcija, turi 2 tipus:
***ArrayList*** -> greitas išrinkimas pagal indeksą

***LinkedList*** -> greitas įterpimas ir šalinimas
List<Tipas> listoVardas = new ArrayList<>();
***pvz:***
List<String> names = new ArrayList<>();
*duomenų paėmimas:*
names.get(0)//pirmas įrašas
*duomenų pridėjimas:*
name.add("Jonas");
***Set:***
//Setas, saugo tik unikalius įrašus, turi 3 tipus :
**HashSet**-> nėra aiškaus rikiavimo tipo
**LinkedHashSet** -> kokia tvarka sudėta tokia ir grąžins
**TreeSet** - > surikiuos didėjimo tvarka
Set<Tipas> vardas = new HashSet<>();
***pvz:***
Set<Integer> ages = new TreeSet<>();
*duomenų pridėjimas:*
ages.add(55);
*duomenų paėmimas:*
Iš set kolekcijos negalime paimti duomenų pagal indexą!
***Map:***
//Map saugo raktas-reikšmė įrašus. Raktas turi būti unikalus
//reikšmės gali kartotis, turi 3 tipus:
**HashMap** -> nėra aiškaus rikiavimo
**LinkedHashMap** - > kaip sudėta taip ir grąžins
**TreeMap** - >Surikiuoja pagal raktą didėjimo tvarka
Map<raktoTipas, reikšmėsTipas> mapVardas = new TreeMap<>();
***pvz:***
Map<Integer, String> map = new TreeMap<>();
*duomenų įdėjimas:*
map.put(12, "Jonas");
*duomenų išgavimas:*
map.get(12);

---

**SQL**

***Duomenų bazės sukūrimas/ištrynimas***
CREATE DATABASE duomenuBazesVardas
DROP DATABASE duomenuBazesVardas
pvz: CREATE DATABASE KSC
***Lentelės sukūrimas/ištrynimas***
CREATE TABLE lentelesVardas(stulpelioVardas, tipas)//
pvz:
CREATE TABLE students(id int not null auto_increment, name varchar(50) not null, surname varchar(50) not null, primary key(id))
//auto_increment - visada užpildomas po viena pridedama
//primary key - nurodo, kad stulpelis yra unikalus
***Duomenų paėmimas***
SELECT * FROM lentelesVardas
pvz: SELECT * FROM students;
***Duomenų įrašymas***
pvz: INSERT INTO lentelesVardas(stupVardas, sulpVardas) VALUES ('reiksme', reiksme)// jei stringas reikia '', jei ne //nereikia
***Duomenų ištrynimas***
DELETE FROM lentelesVardas
pvz: DELETE FROM students //ištrins visus įrašus iš lentelės, //naudokite filtrą, jei reikia ištrinti tik tam tikrus įrašus
***Duomenų atnaujinimas***
UPDATE lentelesVardas SET stulpelioVardas=reiksme
*pvz:* UPDATE students SET name = 'PETRAS'//naudokite filtrą //kitaip bus atnaujinami visi irašai. Jei reikia atnaujinti kelis //stulpelius, jie skiriami per kablelį
***Duomenų filtravimas***
*WHERE*
//Naudojami, kai norisi išrinkti, ištrinti, atnaujinti ne visus //įrašus ,o tik tuos kurių reikia, tam naudojama WHERE
pvz: SELECT * FROM students WHERE name = 'Petras';
DELETE FROM students WHERE id = 1;
*IN*
//naudojamas nurodyti vieną iš daugelio tinkamumų.
//Naudojamas SELECT, INSERT, DELETE, UPDATE...
SELECT * FROM students WHERE stulpelioVardas IN (reiksme, reiksme)
*LIKE*
//Naudojamas, kai nėra žinomas tikslus parametras
SELECT * FROM lentelesVardas WHERE stulepelioVardas LIKE '%kintamasis%'
//%-nurodo, kad nėra svarbi pradžia jei yra pradžioje arba //pabaiga jei yra pabaigoje, jei abiejuose tai tarp
SELECT * FROM students WHERE name LIKE '%as';
*OR*
//išrenka duomenis, kurie yra lygūs ieškomai reikšmei arba //kitai ieškomai reikšmei
SELECT * FROM lentelesVardas WHERE stulpVardas = 'val' OR stulpVardas = 'val'
pvz:
SELECT * FROM students WHERE name ='Andrius' or name ='Petras'
*Duomenų surikiavimas*
//ORDER BY turi eiti paskutiniai, jiems nereikia WHERE
//ASC- surikiuoja nuo mažiausio iki didžiausio
//DASC - nuo didžiausio iki mažiausio
SELECT * FROM lentelesVardas ORDER BY stulpVardas ASC
pvz:
SELECT * FROM students ORDER BY name ASC
***Duomenų apjungimas***
//Naudojami apjungti duomenis iš kelių lentelių. Turi būti //laukas, pagal kuriį sujungsim
LEFT JOIN
//jungiama pagal kairę lentelę, jei nėra įrašų dešinėje //lentelėje jie užpildome null
SELECT s.*, sa.* FROM lentelesVardas s
LEFT JOIN lentelesVardas sa ON s.stulpVardas = sa.stulpVardas
pvz:
SELECT s.*, sa.* FROM students s
LEFT JOIN studentsAddress sa ON s.id = sa.id
RIGHT JOIN
// jungiami įrašai pagal dešinę lentelę. Jei nėra įrašų kairioje //lentelėje užpildo null
SELECT s.*, sa.* FROM lentelesVardas s
RIGHT JOIN lentelesVardas sa ON s.stulpVardas = sa.stulpVardas
pvz:
SELECT s.*, sa.* FROM students s
RIGHT JOIN studentsAddress sa ON s.id = sa.id
INNER JOIN
// išrenka tik tuos įrašus, kurie yra tik abiejuose lentelėse
SELECT s.*, sa.* FROM lentelesVardas s
INNER JOIN lentelesVardas sa ON s.stulpVardas = sa.stulpVardas
pvz:
SELECT s.*, sa.* FROM students s
INNER JOIN studentsAddress sa ON s.id = sa.id