Tehtäväsetti 1.

N3303, Joonas Kangaskoski

**Tehtävä 1.**

* Tehtävien teko hyväksyttävästi, palautus ajallaan PDF:nä. Lopputentin hyväksytysti suorittaminen
* Lopputentti koostuu teoriatestistä ja ohjelmointitehtävistä. Molemmat tulee olla hyväksytysti suoritettu. Pakolliset tehtävät tulee olla palautettu ennen tenttiä, jotta voi osallistua.
* Itsenäistä opiskelua ja asiaan perehtymistä, myös vapaa-ajalla.

**Tehtävä 2.**

Pystytettin tunnilla.

**Tehtävä 3.**

const nimi = 'Pekka'; *// vakio ei tarvitse tyypittää, tyyppinä "pekka"*

let luku; *// tyyppinä any, huono juttu, täytyy tyypittää*

luku = 42;

let ulkonaSataa = false; *// tyyppinä boolean, tyypittää automaattisesti, tunnistaa arvosta.*

const lampotilat = []; *// taas tyyppinä any[], täytyy tyypittää muotoon number[]*

lampotilat.push(15.3, 18.6, 8.7, 10.1, 9.8, 6.6, 7.5);

**Tehtävä 4.**

let lampotila: string | number

function saatila(*l*:number) {

  return *l* === 0 ? 'Suojakeli' : *l* < 0 ? 'Pakkasta' : 'Lämpöasteita';

}

*// testaa tätä stringin lisäksi numerolla*

lampotila = '13';

lampotila = -20

const numeroLampotila = parseInt(lampotila.toString());

console.log(saatila(numeroLampotila));

**Tehtävä 5.**

type Lampotila =  string | number;

**Tehtävä 6.**

function kertotaulu(*x*:number, *y*:number) {

  const kertsi:number[] = [];

  for (let i = 1; i <= *y*; i++) {

    kertsi.push(i \* *x*);

  }

  return kertsi;

}

const keskiarvoMerkkijono = (*laskettava*:string, *numerot*:number[]) => {

  let summa = 0;

*numerot*.forEach((*x*:number) => (summa += *x*));

  return `Keskiarvo (${*laskettava*}) on ${summa / *numerot*.length}`;

};

let kolmosenKertsi = kertotaulu(3, 10);

console.log(keskiarvoMerkkijono('ikä', kolmosenKertsi));

*// tyypitetään anyt, niin palautus selkeämpi, palautuksessa mukana stringiä sekä numeroa*

**Tehtävä 7.**

const summaHomma = function (*taul*:number[], cb:(*k*:number)=>number) {

*// tästä yksinkertaistettin turha iffittely tarkistus pois, koska sen täytyy olla jo funktio sillä tyypitetty aiemmin*

    let tulos = 0;

    for (const alkio of *taul*) {

      tulos += cb(alkio);

    }

    return tulos;

};

const taulukko = [1, 2, 3, 4, 5];

*// laske pelkkä summa*

console.log(summaHomma(taulukko, (*luku*) => *luku*));

*// laske summa neliöistä, kakkosen potensseista*

console.log(summaHomma(taulukko, (*luku*) => *luku* \* *luku*));

**Tehtävä 8.**

type Auto =  {

  merkki: string;

  kaynnistyy:boolean;

  ovienMaara:number;

}

type Mopo = {

  merkki:string;

  kaynnistyy:boolean;

}

type Menopeli = Auto | Mopo

let v1, v2: Menopeli;

v1 = {

  merkki: 'Trabant',

  kaynnistyy: true,

  ovienMaara: 4

}

v2 = {

  merkki:'Raisu',

  kaynnistyy:false *// arvolla true raisu sanoo wrooomm!!*

}

function aja(*x*:Menopeli){

if (!*x*.kaynnistyy ){

  console.log(`${*x*.merkki} ei käynnisty`)

}else {

  if ("ovienMaara" in *x*){

    for(let i = 0; i<*x*.ovienMaara;i++){

      console.log('Slap!')

    }

    console.log(`${*x*.merkki} sanoo wroom` )

} else {

  console.log('Raisu sanoo wroom')

}

}

}

aja(v2)

aja(v1)

**Tehtävä 9.**

type Kaara = {

  merkki: string;

  kaynnistyy:boolean;

  ovienMaara: number;

  kaasuta:(*a*:number)=>void;

}

let auto:Kaara = {

  merkki: 'Trabant',

  kaynnistyy: true,

  ovienMaara: 4,

  kaasuta: function (*lkm*) {

    for (let i = 0; i < *lkm*; i++) {

      console.log('Wroom');

    }

  },

};

auto.kaasuta(5);

**Tehtävä 10.**

interface Kaara  {

  merkki: string;

  kaynnistyy:boolean;

  ovienMaara: number;

  kaasuta:(*a*:number)=>void;

}

let auto:Kaara = {

  merkki: 'Trabant',

  kaynnistyy: true,

  ovienMaara: 4,

  kaasuta: function (*lkm*)  {

    for (let i = 0; i < *lkm*; i++) {

      console.log('Wroom');

    }

  },

};

auto.kaasuta(5);

**Tehtävä 11.**

interface Kaara  {

  merkki: string;

  kaynnistyy:boolean;

  ovienMaara: number;

  ratinSijainti: 'Oikea' | 'Vasen';

  kaasuta:(*a*:number)=>void;

}

let auto:Kaara = {

  merkki: 'Trabant',

  kaynnistyy: true,

  ovienMaara: 4,

  ratinSijainti: 'Vasen',

  kaasuta: function (*lkm*)  {

    for (let i = 0; i < *lkm*; i++) {

      console.log('Wroom');

    }

  },

};

auto.kaasuta(5);

console.log(auto.ratinSijainti)

**Tehtävä 12.**

interface Kaara  {

  merkki: string;

  kaynnistyy:boolean;

  ovienMaara: number;

  ratinSijainti: 'Oikea' | 'Vasen';

  katsastustiedot:[Date | string, string, boolean][]; *// tuple määrittely*

  kaasuta:(*a*:number)=>void;

}

let auto:Kaara = {

  merkki: 'Trabant',

  kaynnistyy: true,

  ovienMaara: 4,

  ratinSijainti: 'Vasen',

  katsastustiedot:[

*// annetaan taulokolle sisältöä*

    [new Date("2021-05-16").toLocaleDateString(),'Jyväskylänkatsastus', false],

    [new Date("2020-03-12").toLocaleDateString(),'Mikkelin rompepaja', true],

     [new Date("2019-06-10").toLocaleDateString(),'Keski-Suomen auto',true],

     [new Date("2018-02-6").toLocaleDateString(),'Keski-Suomen auto',true]

  ],

  kaasuta: function (*lkm*)  {

    for (let i = 0; i < *lkm*; i++) {

      console.log('Wroom');

    }

  },

};

const katsastus = (*a*:[Date | string ,string,boolean][]) => {

*a*.forEach((*x*) => {

      if (*x*[2]){

        console.log(`${*x*[0]}  paikassa ${*x*[1]} ja tuloksena Hyväksytty`)

      }else{

        console.log(`${*x*[0]}  paikassa ${*x*[1]} ja tuloksena Hylätty`)

      }

    })

  }

auto.kaasuta(5);

console.log(auto.ratinSijainti)

console.log(auto.katsastustiedot)

katsastus(auto.katsastustiedot)

**Tehtävä 13.**

interface Ihminen {

  etunimi:string;

  sukunimi:string;

  ika:number;

  taidot?:string[];

  taitotaso?():number;

}

const pekka: Ihminen = {

  etunimi: 'Pekka',

  sukunimi: 'Pekkanen',

  ika: 30,

  taidot: ['ohjelmointi', 'ahmiminen'],

  taitotaso() {

    if(this.taidot){

    return this.taidot.length;

  }else {

    return 0

  }

  },

};

function tiedot(*hlo*: Ihminen): void {

  console.log(

    `Henkilö on ${*hlo*.etunimi} ${*hlo*.sukunimi} ja hänellä on ikää ${*hlo*.ika} vuotta`

  );

}

tiedot(pekka);

**Tehtävä 14.**

class Henkilo {

  public etunimi: string;

  public sukunimi: string;

  public kengannumero: number;

  public svuosi: number;

  constructor(*etunimi*:string, *sukunimi*:string,*kengannumero*:number,*svuosi*:number){

    this.etunimi = *etunimi*;

    this.sukunimi = *sukunimi*;

    this.kengannumero = *kengannumero*;

    this.svuosi = *svuosi*;

  }

  faktat(){

    console.log(`Nimeni on ${this.etunimi} ${this.sukunimi} ja kengännumeroni on ${this.kengannumero}`)

  }

}

const Pekka = new Henkilo('Pekka','Pekkanen', 44, 1980)

Pekka.faktat()

console.log(Pekka.svuosi)

Pekka.svuosi = 2000

console.log(Pekka.svuosi)

**Tehtävä 15.**

class Henkilo {

  public etunimi: string;

  public sukunimi: string;

  public kengannumero: number;

  private svuosi: number;

  constructor(*etunimi*:string, *sukunimi*:string,*kengannumero*:number,*svuosi*:number){

    this.etunimi = *etunimi*;

    this.sukunimi = *sukunimi*;

    this.kengannumero = *kengannumero*;

    this.svuosi = *svuosi*;

  }

  faktat(){

    console.log(`Nimeni on ${this.etunimi} ${this.sukunimi} ja kengännumeroni on ${this.kengannumero}`)

  }

  get ika(){

        return new Date().getFullYear() - this.svuosi

      }

  set ika(*x:*number){

        this.svuosi = *x*

      }

}

const Pekka = new Henkilo('Pekka','Pekkanen', 44, 1980)

Pekka.faktat()

console.log(Pekka.ika)

Pekka.ika = 1990

console.log(Pekka.ika)

**Tehtävä 16.**

class Kirja {

*// tyypitetään muuttujat.*

  public nimi:string;

  public kustantaja:string;

  public julkaisuvuosi:number;

  public sivumaara:number;

  public tekijat:string[];

*// jokaisella luokalla täytyy olla konstruktori... luodaan se.*

  constructor(*nimi*:string, *kustantaja*:string, *julkaisuvuosi*:number, *sivumaara*:number, *tekijat*:string[]){

    this.nimi = *nimi*;

    this.kustantaja = *kustantaja*;

    this.julkaisuvuosi = *julkaisuvuosi*;

    this.sivumaara = *sivumaara*;

    this.tekijat = *tekijat*;

  }

*// metodi tulostaa kirjan julkaisutiedot*

  julkaisutiedot(){

    console.log(`Kustantaja ${this.kustantaja} julkaisi vuonna ${this.julkaisuvuosi} kirjan ${this.nimi}`)

  }

*// metodi tulostaa kirjan tekijälistan*

  tekijaLista(){

    console.log('Kirjan tekijät: ')

    for(let i = 0; i < this.tekijat.length; i++){

      console.log(this.tekijat[i] + ',')

    }

  }

*// metodi lisää parametrin kirjantekijöihin*

  lisaaTekija(*a*:string) {

    this.tekijat.push(*a*)

  }

}

*// luodaan uusi kirja*

const morko = new Kirja('Mörön suuri seikkailu', 'WSOY', 2008, 110, ['Martti Kaunokirjoitus', 'Kalle Konekääntäjä'] )

*// tulostetaan julkaisutiedot*

morko.julkaisutiedot()

*// tulostetaan tekijalista*

morko.tekijaLista()

*// lisätään kirjoittaja*

morko.lisaaTekija('Olli Oikolukija')

*// lisäyksen tarkistus*

morko.tekijaLista()

**Tehtävä 17.**

class Kirja {

*// tyypitetään muuttujat.*

  public nimi:string;

  public kustantaja:string;

  public julkaisuvuosi:number;

  public sivumaara:number;

  public tekijat:string[];

  private static sivujaYht:number = 0;

*// jokaisella luokalla täytyy olla konstruktori... luodaan se.*

  constructor(*nimi*:string, *kustantaja*:string, *julkaisuvuosi*:number, *sivumaara*:number, *tekijat*:string[]){

    this.nimi = *nimi*;

    this.kustantaja = *kustantaja*;

    this.julkaisuvuosi = *julkaisuvuosi*;

    this.sivumaara = *sivumaara*;

    this.tekijat = *tekijat*;

    Kirja.sivujaYht = Kirja.sivujaYht + this.sivumaara;

  }

*// metodi tulostaa kirjan julkaisutiedot*

  julkaisutiedot(){

    console.log(`Kustantaja ${this.kustantaja} julkaisi vuonna ${this.julkaisuvuosi} kirjan ${this.nimi}`)

  }

*// metodi tulostaa kirjan tekijälistan*

  tekijaLista(){

    console.log('Kirjan tekijät: ')

    for(let i = 0; i < this.tekijat.length; i++){

      console.log(this.tekijat[i] + ',')

    }

  }

*// metodi lisää parametrin kirjantekijöihin*

  lisaaTekija(*a*:string) {

    this.tekijat.push(*a*)

  }

*// luodaan staattinen metodi joka palauttaa kirjojen sivut yhteensä*

  static sivujaKirjoissa():number{

    return Kirja.sivujaYht

  }

}

*// luodaan uusi kirja*

const peppi = new Kirja('Peppi Pitkätossu', 'WSOY', 2008, 85, ['Ule Bergström', 'Sven Hassel'] )

const morko = new Kirja('Mörön suuri seikkailu', 'WSOY', 2008, 110, ['Martti Kaunokirjoitus', 'Kalle Konekääntäjä'] )

*// tulostetaan julkaisutiedot*

morko.julkaisutiedot()

*// tulostetaan tekijalista*

morko.tekijaLista()

*// lisätään kirjoittaja*

morko.lisaaTekija('Olli Oikolukija')

*// lisäyksen tarkistus*

morko.tekijaLista()

*// tulostetaan kirjojen sivut*

console.log(Kirja.sivujaKirjoissa())

**Tehtävä 18.**

class Kirja {

*// tyypitetään muuttujat.*

  public nimi:string;

  public kustantaja:string;

  public julkaisuvuosi:number;

  private sivumaara:number;

  public tekijat:string[];

  private static sivujaYht:number = 0;

*// jokaisella luokalla täytyy olla konstruktori... luodaan se.*

  constructor(*nimi*:string, *kustantaja*:string, *julkaisuvuosi*:number, *sivumaara*:number, *tekijat*:string[]){

    this.nimi = *nimi*;

    this.kustantaja = *kustantaja*;

    this.julkaisuvuosi = *julkaisuvuosi*;

    this.sivumaara = *sivumaara*;

    this.tekijat = *tekijat*;

    Kirja.sivujaYht = Kirja.sivujaYht + this.sivumaara;

  }

*// metodi tulostaa kirjan julkaisutiedot*

  julkaisutiedot(){

    console.log(`Kustantaja ${this.kustantaja} julkaisi vuonna ${this.julkaisuvuosi} kirjan ${this.nimi}`)

  }

*// metodi tulostaa kirjan tekijälistan*

  tekijaLista(){

    console.log('Kirjan tekijät: ')

    for(let i = 0; i < this.tekijat.length; i++){

      console.log(this.tekijat[i] + ',')

    }

  }

*// metodi lisää parametrin kirjantekijöihin*

  lisaaTekija(*a*:string) {

    this.tekijat.push(*a*)

  }

*// luodaan staattinen metodi joka palauttaa kirjojen sivut yhteensä*

  static sivujaKirjoissa():number{

    return Kirja.sivujaYht

  }

*// setteri jolla muokataan sivumäärää ja päivitetään sivuYht muuttujaa.*

   set muokkaaSivuMaaraa(*x*:number){

   if (*x* > 0){

    Kirja.sivujaYht = Kirja.sivujaYht - this.sivumaara

    this.sivumaara = *x*

    Kirja.sivujaYht = Kirja.sivujaYht + this.sivumaara;

   } else {

     console.log('Et voi syöttää miinusmerkkisiä sivumääriä')

   }

  }

}

*// luodaan uusi kirja*

const peppi = new Kirja('Peppi Pitkätossu', 'WSOY', 2008, 85, ['Ule Bergström', 'Sven Hassel'] )

const morko = new Kirja('Mörön suuri seikkailu', 'WSOY', 2008, 110, ['Martti Kaunokirjoitus', 'Kalle Konekääntäjä'] )

*// tulostetaan julkaisutiedot*

morko.julkaisutiedot()

*// tulostetaan tekijalista*

morko.tekijaLista()

*// lisätään kirjoittaja*

morko.lisaaTekija('Olli Oikolukija')

*// lisäyksen tarkistus*

morko.tekijaLista()

*// tulostetaan kirjojen sivut*

console.log(Kirja.sivujaKirjoissa())

morko.muokkaaSivuMaaraa = 200;

console.log(Kirja.sivujaKirjoissa())

**Tehtävä 19.**

*// tehdään luokasta eläkeläinen luokan henkilo aliluokka*

class Elakelainen extends Henkilo{

  private elakevuodet:number;

  constructor(

*etunimi*:string,

*sukunimi*:string,

*kengannumero*:number,

*svuosi*: number,

*elakevuodet*:number,

  ){

*// tuodaan superilla äitiluokan muuttujat*

*super*(*etunimi*,*sukunimi*,*kengannumero*,*svuosi*)

    this.elakevuodet = *elakevuodet*

  }

  elakkeellaVuodesta (){

   return new Date().getFullYear() - this.elakevuodet

  }

  elakeIka() {

    return this.elakkeellaVuodesta() - this.svuosi

  }

}

**Tehtävä 20.**

class Aanikirja extends Kirja{

*// omat muuttujat*

  public lukija:string;

  public pituusMin:number;

*// konstruktori, sisältäen myös äitiluokan muuttujat*

  constructor(*nimi*:string, *kustantaja*:string, *julkaisuvuosi*:number, *sivumaara*:number, *tekijat*:string[], *lukija*:string, *pituusMin*:number){

*// superilla äitiluokan tiedot*

*super*(*nimi*, *kustantaja*, *julkaisuvuosi*, *sivumaara*, *tekijat*)

    this.lukija = *lukija*;

    this.pituusMin = *pituusMin*;

  }

*// syrjäytetään äiti luokan julkaisutiedot metodi.*

  julkaisutiedot(){

    console.log(`Kustantaja ${this.kustantaja} julkaisi äänikirjan ${this.nimi}, jonka pituus on `+ this.pituusMin +' minuuttia')

  }

  }

*// luodaan äänikirja*

const remes = new Aanikirja('Kotka', 'Tammela', 2019, 300,[ 'Ilkka Remes'], ' Pertti Salovaara', 120)

**Tehtävä 21.**

abstract class Elain {

  constructor(public *lajinimi*:string){}

  abstract aantele():void;

}

class Kissa extends Elain{

  constructor(public *nimi*:string, public *ika*:number, public *vari*:string[],*lajinimi*:string){

*super*(*lajinimi*)

  }

  aantele(){

    console.log('Miuaauuuuu')

  }

}

class Papukaija extends Elain{

  constructor( public *nimi*:string, public *ika*:number, *lajinimi*:string){

*super*(*lajinimi*)

  }

  aantele() {

      console.log(`Kraah! nimeni on ${this.nimi}`)

  }

}

const kisu = new Kissa('Peppi', 10, ['Musta','Valkea'], 'Kissa')

kisu.aantele()

const kaija = new Papukaija('Maisteri', 4, 'Papukaija')

kaija.aantele()

**Tehtävä 22.**

abstract class Teos {

  constructor(public *nimi*: string, public *kustantaja*: string, public *julkaisuvuosi*: number, public *tekijat*: string[]) { }

  abstract julkaisutiedot(): void;

  tekijaLista() {

    console.log('Kirjan tekijät: ');

    for (let i = 0; i < this.tekijat.length; i++) {

      console.log(this.tekijat[i] + ',');

    }

  }

  lisaaTekija(*a*: string) {

    this.tekijat.push(*a*);

  }

}

class Kirja extends Teos {

*// tyypitetään muuttujat.*

  private static sivujaYht: number = 0;

  constructor(*nimi*: string, *kustantaja*: string, *julkaisuvuosi*: number, private *sivumaara*: number, *tekijat*: string[]) {

*super*(*nimi*, *kustantaja*, *julkaisuvuosi*, *tekijat*);

    Kirja.sivujaYht = Kirja.sivujaYht + this.sivumaara;

  }

*// metodi tulostaa kirjan julkaisutiedot*

  julkaisutiedot() {

    console.log(`Kustantaja ${this.kustantaja} julkaisi vuonna ${this.julkaisuvuosi} kirjan ${this.nimi}`);

  }

*// luodaan staattinen metodi joka palauttaa kirjojen sivut yhteensä*

  static sivujaKirjoissa(): number {

    return Kirja.sivujaYht;

  }

*// setteri jolla muokataan sivumäärää ja päivitetään sivuYht muuttujaa.*

  set muokkaaSivuMaaraa(*x*: number) {

    if (*x* > 0) {

      Kirja.sivujaYht = Kirja.sivujaYht - this.sivumaara;

      this.sivumaara = *x*;

      Kirja.sivujaYht = Kirja.sivujaYht + this.sivumaara;

    } else {

      console.log('Et voi syöttää miinusmerkkisiä sivumääriä');

    }

  }

}

class Aanikirja extends Teos {

*// konstruktori, sisältäen myös äitiluokan muuttujat*

  constructor(

*nimi*: string,

*kustantaja*: string,

*julkaisuvuosi*: number,

*tekijat*: string[],

    public *lukija*: string,

    public *pituusMin*: number

  ) {

*// superilla äitiluokan tiedot*

*super*(*nimi*, *kustantaja*, *julkaisuvuosi*, *tekijat*);

  }

*// syrjäytetään äiti luokan julkaisutiedot metodi.*

  julkaisutiedot() {

    console.log(

      `Kustantaja ${this.kustantaja} julkaisi äänikirjan ${this.nimi}, jonka pituus on ` + this.pituusMin + ' minuuttia'

    );

  }

}

*// luodaan äänikirja*

const remes = new Aanikirja('Kotka', 'Tammela', 2019, ['Ilkka Remes'], ' Pertti Salovaara', 120);

*// luodaan uusi kirja*

const peppi = new Kirja('Peppi Pitkätossu', 'WSOY', 2008, 85, ['Ule Bergström', 'Sven Hassel']);

const morko = new Kirja('Mörön suuri seikkailu', 'WSOY', 2008, 110, ['Martti Kaunokirjoitus', 'Kalle Konekääntäjä']);

*// tulostetaan julkaisutiedot*

morko.julkaisutiedot();

morko.tekijaLista();

remes.julkaisutiedot();

remes.tekijaLista();

**Tehtävä 23.**

interface IPerusFaktat{

  laji:string;

  svuosi:number;

}

class Koira implements IPerusFaktat{

  public laji: string;

  public rotu:string;

  public nimi:string

  public svuosi: number;

  constructor(*laji*:string,*rotu*:string,*nimi*:string,*svuosi*:number){

    this.laji = *laji*

    this.rotu = *rotu*

    this.nimi = *nimi*

    this.svuosi = *svuosi*

  }

  kuolaa(){

    console.log('Sluurrpp dip dip dip')

  }

}

class Kilpikonna implements IPerusFaktat{

    constructor(public *laji*:string,  public *svuosi*:number, public *vari*:string){

    }

  }

  const musti = new Koira('koira','bernie paimenkoira','Musti',2012 )

  musti.kuolaa()

  const turtle = new Kilpikonna('Turtle', 1990, 'vihreä')

**Tehtävä 24.**

interface IPerusFaktat{

  laji:string;

  svuosi:number;

}

interface ILiikkuva{

  liikkumisaani:string;

  liiku:(*a*?:number)=>void;

}

interface IAanteleva extends IPerusFaktat{

  aani:string;

  aantele:()=>string;

}

type LiikkuuJaAantelee = ILiikkuva & IAanteleva

class Koira implements LiikkuuJaAantelee{

  public laji: string;

  public rotu:string;

  public nimi:string

  public svuosi: number;

  public aani:string

  public liikkumisaani: string;

  constructor(*laji*:string,*rotu*:string,*nimi*:string,*svuosi*:number,*liikkumisaani*:string,*aani*:string){

    this.laji = *laji*

    this.rotu = *rotu*

    this.nimi = *nimi*

    this.svuosi = *svuosi*

    this.aani = *aani*

    this.liikkumisaani = *liikkumisaani*

  }

  kuolaa(){

    console.log('Sluurrpp dip dip dip')

  }

  liiku(*x*:number = 1){

    for (let i = 0; i < *x*; i++) {

      console.log(this.liikkumisaani);

  }

}

 aantele(){

  return this.aani

  }

}

class Kilpikonna implements IPerusFaktat,ILiikkuva{

    constructor(public *laji*:string,  public *svuosi*:number, public *vari*:string,public *liikkumisaani*:string){

    }

    liiku(){

      console.log(this.liikkumisaani)

    }

  }

  const musti = new Koira('koira','bernie paimenkoira','Musti',2012,'skrips skraps','äyhöyh wuh wuh' )

  musti.kuolaa()

  const turtle = new Kilpikonna('Turtle', 1990, 'vihreä','önts önts önts')

musti.aantele()

musti.liiku()

**Tehtävä 25.**

let nimet = opiskelijat.map(*x* => {

*// jos vasen operandi on null tai undefined palautetaan 'ei tiedossa'*

  return *x*.tiedot?.nimi ?? 'Ei tiedossa'

})

console.log(nimet)

let kaupungit = opiskelijat.map(*x* => {

*// jos vasen operandi on null tai undefined palautetaan 'ei tiedossa'*

  return *x*.tiedot?.kaupunki ?? 'Ei tiedossa'

})

console.log(kaupungit)