JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

Ohjelmointi 2 – Luento 3

Olio-ohjelmoinnin perusteet, osa 2:

Perintä, polymorfismi, abstrakti luokka

Luennon sisältö

- Perintä
- Polymorfismi
- Abstrakti luokka

Pari muistutusta

- Denis poissa tämän ja ensi viikon
- Hyödyntäkää ohjauksia

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

Perintä

Perintä

- Saman tai saman tapaisen koodin kirjoittamista uudelleen tulee välttää
- ”Don’t Repeat Yourself” (Hunt & Thomas 1999)
- Olioilla voi olla yhteisiä piirteitä, joita voi olla hyödyllistä yhdistellä myös koodin tasolla

```
void main()
{
    Kissa kissa = new Kissa("Miuku", "Siamilainen");
    Koira koira = new Koira("Rekku", "Bulldoggi");
    Marsu marsu = new Marsu("Nappi", "Agouti");
}
```

```java
public class Kissa {
    private final String nimi;
    private final String rotu;

    public Kissa(String nimi, String rotu) {
        this.nimi = nimi;
        this.rotu = rotu;
    }
}
```

```java
public class Koira {
    private final String nimi;
    private final String rotu;

    public Koira(String nimi, String rotu) {
        this.nimi = nimi;
        this.rotu = rotu;
    }
}
```

```java
public class Hamsteri {
    private final String nimi;
    private final String rotu;

    public Hamsteri(String nimi, String rotu) {
        this.nimi = nimi;
        this.rotu = rotu;
    }
}
```

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

```java
public class Kissa {
    private final String nimi;
    private final String rotu;

    public Kissa(String nimi, String rotu) {
        this.nimi = nimi;
        this.rotu = rotu;
    }

    public String getNimi() {
        return nimi;
    }

    public String getRotu() {
        return rotu;
    }
}
```

```java
public class Koira {
    private final String nimi;
    private final String rotu;

    public Koira(String nimi, String rotu) {
        this.nimi = nimi;
        this.rotu = rotu;
    }

    public String getNimi() {
        return nimi;
    }

    public String getRotu() {
        return rotu;
    }
}
```

```java
public class Hamsteri {
    private final String nimi;
    private final String rotu;

    public Hamsteri(String nimi, Strin
        this.nimi = nimi;
        this.rotu = rotu;
    }

    public String getNimi() {
        return nimi;
    }

    public String getRotu() {
        return rotu;
    }
}
```

Perintä

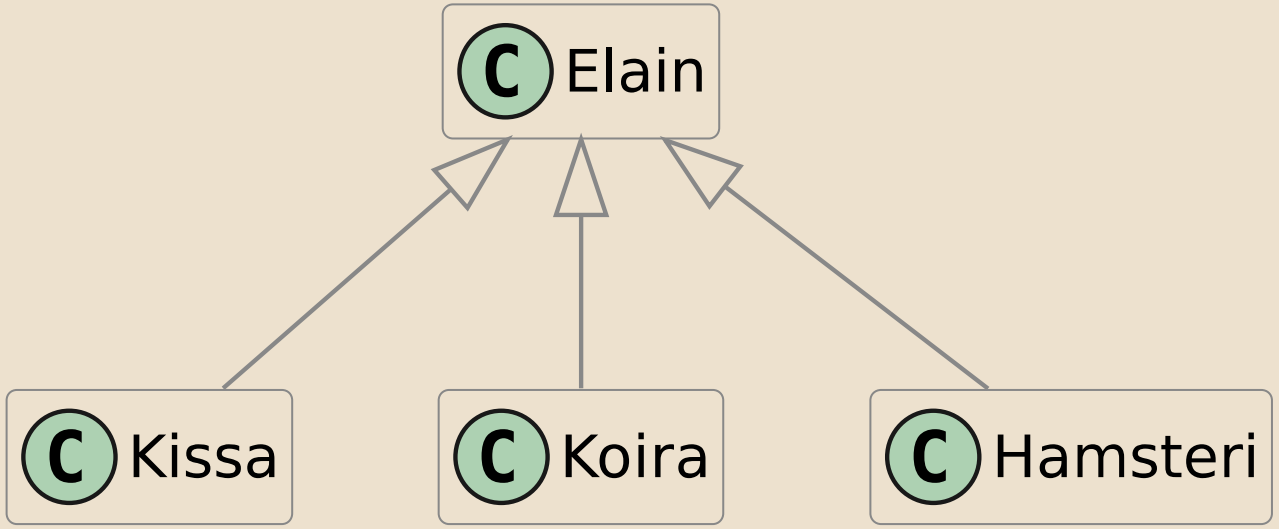
- Perintä (engl. *inheritance*) mahdollistaa luokkien uudelleenkäytön ja hierarkkisen rakenteen luomisen
- *is-a*-suhde: Opiskelija on Henkilö, Kissa on Eläin, PhysicsObject on GameObject (Jypeli)
- Mahdollistaa uuden luokan luomisen olemassa olevasta luokasta
- Perivää luokkaa kutsutaan aliluokaksi, perittävää luokkaa yliluokaksi
- Aliluokka saa yliluokan ei-yksityiset ominaisuudet ja metodit
- Java-kielessä perintä toteutetaan avainsanalla `extends`

Muodostaja ja super

- Aliluokasta voidaan kutsua yliluokan muodostajia avainsanalla `super`

Demo

Perintähierarkia UML-kaaviona



JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

Polymorfismi

Polymorfismi

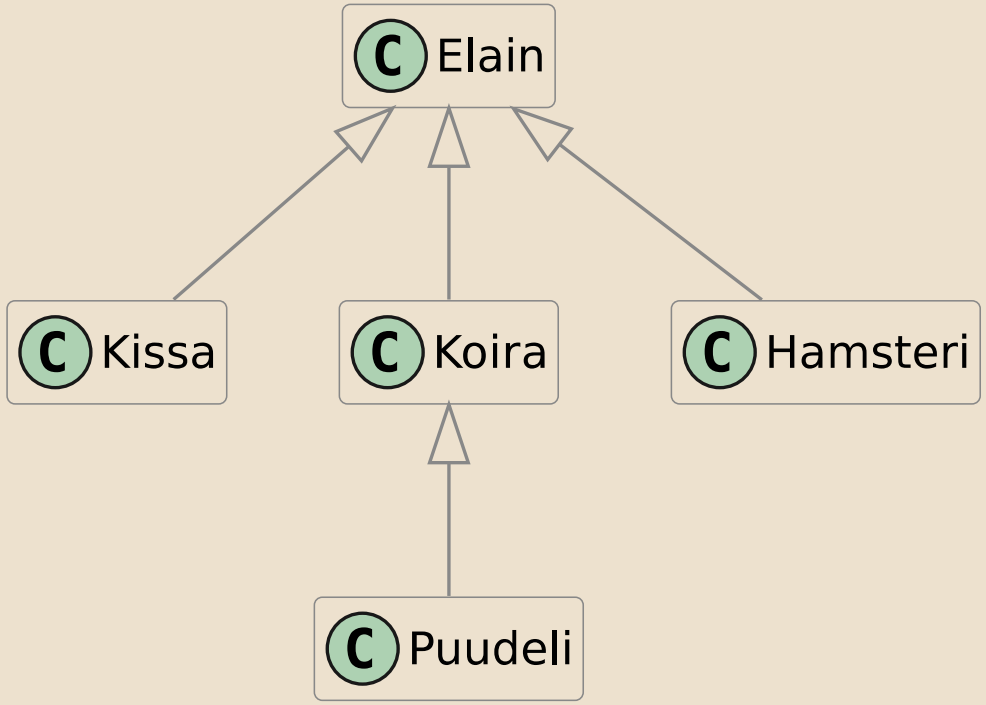
- Polymorfismi on periaate, joka tarkoittaa tavoitetta käsitellä eri tyyppisiä olioita saman yleisen tyypin kautta kuitenkin niin, että kukin olio käyttäytyy sille ominaisella tavalla
- Yleinen tyyppi (ts. ylätyyppi) voi olla luokka tai rajapinta
- Kun metodia kutsutaan, päätös siitä, mikä metodi tosiasiallisesti suoritetaan, tehdään ajon aikana olion todellisen tyypin perusteella (engl. *dynamic binding*)

Metodit ja super

- Aliluokan metodeissa voidaan viitata yliluokan metodiin eksplisiittisesti avainsanalla `super`

Demo

Perintähierarkia UML-kaaviona



Polymorfismin hyötyjä

- Koodin yleistettävyys, abstraktio. Vertaa:
  - void metodi(Kissa k) *vs.* void metodi (Elain e)
- Kokoelmien hallinta: Yhteisen ylätyypin avulla voimme käsitellä joukkoa yhtenäisesti
  - for (Kissa k : kissat) *vs.* for (Elain e : elain)
- Avoin/suljettu-periaate (”open closed principle”): koodin tulisi olla avoin laajennuksille, mutta suljettu muutoksille
  - if (elain instance of Kissa k) { k.aantele(); } *vs.* elain.aantele();

Huomautus

- Tutustumme tänään polymorfismiin luokkien perinnän näkökulmasta
- Sama ideaa toteutetaan myös rajapintojen yhteydessä seuraavissa osissa

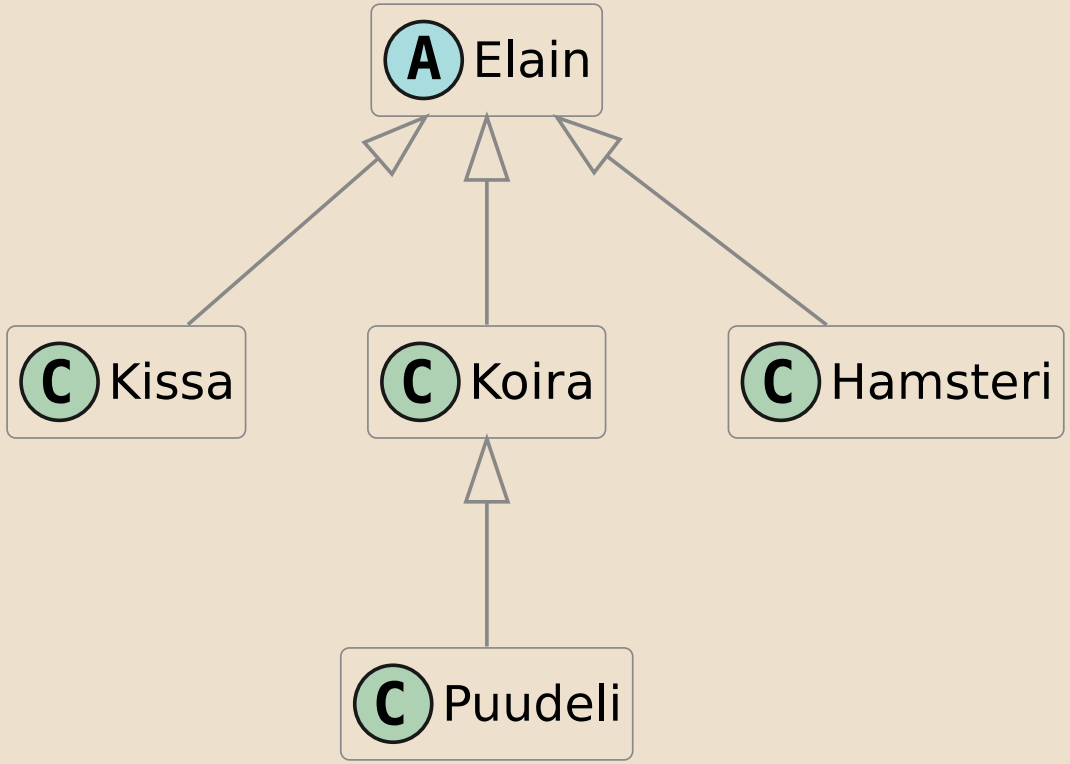
JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

Abstrakti luokka

Abstrakti luokka

- ”Ei-konkreettinen” luokka
- Käytetään yleensä yliluokkana, jonka muut luokat perivät
- Ei voida suoraan tehdä olioinstanssia
- Abstrakti luokka voi sisältää
  - tavallisia metodeja, jotka peritään kuten muutkin
  - abstrakteja metodeja, jotka on määriteltävä aliluokissa
- Abstrakti luokka määritellään avainsanalla `abstract`.

Demo


A Elain
C Kissa
C Koira
C Hamsteri
C Puudeli

Mitä seuraavaksi

- Kyselytunti klo 17 saakka
- Harjoitustehtävät 3