16. Geneerinen ohjelmointi

Sisällys

- Johdanto.
- Geneerisen tyypin määrittely tyyppiparametrin avulla
- Geneerisen tyypin käyttöönotto tyyppiparametrin arvo kiinnittämällä.
- Tyyppiparametrin rajoittaminen.
- Geneerinen tyyppi käännöksessä.
- Raakatyyppi.
- Geneeristen tyyppien yhteensopivuus.
- Geneeriset tyypit ja periytyminen.

Johdanto

- Geneerinen ohjelmointi (generic programming, generics) tarkoittaa muun muassa ohjelmointitekniikkaa, jossa algoritmeja toteutetaan yleisesti sitomatta niitä käsiteltävän tiedon tyyppiin.
  - Tunnetuin geneerisyyden mahdollistava mekanismi lienevät C++-kielen kaavaimet (template).
  - Javaan geneerisyys lisättiin versiossa 1.5.
- Javan geneerisyys tuo ohjelmaan lisää tyyppitarkistuksia ja vähentää tyyppimuunnoksia.
  - Algoritmien yleinen toteutus on Javassa mahdollista myös tyyppien monimuotoisuuden avulla.

Johdanto

- Geneerisyyden tuottamat lisätarkistukset eivät ole käytettävissä Java-ohjelman suorituksen aikana.
  - Geneeriset määreet korvataan tyypeillä käännöksen aikana.
- Javassa geneerisyydestä on apua erityisesti, kun tavoitteena on käyttää kokoelmaluokkia (esimerkiksi *ArrayList*) turvallisemmin.
- Geneerinen ohjelmointi ei ole pakollista ja geneerisyyttä voi käyttää vain osassa ohjelmaa.
  - Java-kääntäjä varoittaa, jos ohjelmassa on osuus, jossa geneerisyyttä käytetään tai jätetään käyttämättä siten, että tuloksena on mahdollisesti vaarallisia lauseita.

Geneerisen tyypin määrittely

- Geneerinen tyyppi on luokka tai rajapinta, jonka määrittelyssä annetaan yksi tai useampi tyyppiparametri kulmasuljeparin <> sisällä tyypin tunnuksen jälkeen.
- Tyyppiparametria voidaan käyttää luokassa esimerkiksi attribuutin ja muuttujan tyyppinä sekä metodin paluuarvon ja parametrin tyyppinä.
- Tyyppiparametrin tunnuksena on yksittäinen suuri kirjain.
  - *E* = alkio (element), *K* = avain (key), *N* = luku (number), *T* = tyyppi (type), *V* = arvo (value).

Geneerisen tyypin määrittely

- // Laatikko, jonka sisällön tyyppi on parametrisoitu.
  ```
  public class Laatikko<T> {
     private T sisältö; ...
  ```

- // Javan rajapinta olioiden suunnattuun vertailuun.
  ```
  public interface Comparable<T> {
     public abstract int compareTo(T o); ...
  ```

- // Javan säiliöluokka avain–arvo-parien säilömiseen.
  ```
  public class HashMap<K,V> ...
  ```

Geneerisen tyyppi käyttöönotto

- Geneerinen tyyppi (luokka tai rajapinta) otetaan käyttöön muualla ohjelmassa kiinnittämällä tyyppiparametrille arvo siten, että geneerisen tyypin tunnuksen esiintymien perään lisätään jonkin viitetyypin tunnus kulmasulkeiden sisässä.
  - Tyyppiparametrin arvon tulee olla sama kaikissa kiinnityksissä.
  - Tyyppiparametrin arvo ei voi valitettavasti olla alkeistyyppi.
- Tehtävä määrää tyyppiparametrin arvon.
  - Geenerisen tyypin käyttö on sitä suoraviivaisempaa mitä kauempana luokkahierarkian juuresta tyyppi on.
    - Tyyppitarkistukset ovat tiukimpia ja tyyppimuunnosten määrä on vähimmillään, kun tyyppi on luokkahierarkian lehtiluokan tyyppi.
    - *Object*-tyyppillä saavutetaan monimuotoisuuteen verrattuna pienimmät edut.

Geneerisen tyyppi käyttöönotto

- // Rakennetaan laatikko, jossa voi olla vain kissoja.
  Laatikko<Kissa> kissanLaatikko = **new** Laatikko<Kissa>(ville);
- // Nisäkäs voi verrata itseensä nisäkkäitä ja jälkeläisiä.
  public abstract class Nisakas **implements**
  Comparable<Nisakas> { …
- Tyyppiparametrin arvo kiinnitetään pääsääntöisesti aina, kun generisoidun tyypin tunnus esiintyy ohjelmassa.
  - Java voi joskus päätellä tyyppiparametrin arvon itse. Tällöin riittää tyhjä kulmasulkeiden pari eli “timantti”.
- // Jälkimmäinen tyyppiparametrin arvo seuraa ensimmäisestä.
  Laatikko<Kissa> kissanLaatikko = **new** Laatikko<>(ville);

Geneerinen laatikko (Laatikko.java)

```
// Laatikkoa mallintava luokka. Laatikko voi sisältää minkä
// tahansa tyyppisen olion, koska olion tyyppi on parametrisoitu.
public class Laatikko<T> {
   // Viite laatikon sisältöön.
   private T sisältö;
   // Metodit.
   public Laatikko(T uusiSisältö) ... {
   public T sisältö() { ...
   public void sisältö(T uusiSisältö) ... {
}
```

Geneerinen laatikko (LaatikkoTesti.java)

```
...
Kissa ville = new Kissa();

// Luodaan laatikko kissoille kiinnittämällä tyyppiparametri
// ja asetetaan Ville laatikkoonsa.
Laatikko<Kissa> kissanLaatikko = new Laatikko<Kissa>(ville);

// Tutkitaan laatikon sisältöä. Tyyppimuunnoksia ei tarvita,
// koska laatikko on varattu kissoille.
Kissa laatikossa = kissanLaatikko.sisältö();

// Alla oleva ei käänny, koska laatikossa voi olla vain kissoja.
kissanLaatikko.sisältö("K.I.S.S.A");
...
```

Monimuotoinen laatikko (Laatikko.java)

```
// Laatikkoa mallintava luokka. Laatikko voi sisältää minkä
// tahansa tyyppisen olion monimuotoisuuden ansiosta:
// attribuutin tyyppi on Object, joka kanssa mikä tahansa
// luokkatyyppi on yhteensopiva.
public class Laatikko {
   // Viite laatikon sisältöön.
   private Object sisältö;
   // Metodit.
   public Laatikko(Object uusiSisältö) ... {
   public Object sisältö() { ...
   public void sisältö(Object uusiSisältö) ... {
}
```

Monimuotoinen laatikko (LaatikkoTesti.java)

```
...
Kissa ville = new Kissa();

// Luodaan laatikko ja asetetaan Ville laatikkoonsa.
Laatikko kissanLaatikko = new Laatikko(ville);

// Tutkitaan laatikon sisältöä. Tyyppimuunnos tarvitaan,
// koska lukuaksessori palauttaa Object-tyyppisen viitteen.
Kissa laatikossa = (Kissa)kissanLaatikko.sisältö();

// Monimuotoiseen laatikkoon voidaan asettaa tarkoituksella
// ja vahingossa muitakin kuin kissaolioita.
kissanLaatikko.sisältö("K.I.S.S.A");
...
```

Tyyppiparametrin rajoittaminen

- Tyyppiparametrin arvoa voidaan rajoittaa geneerisessä tyypissä määrittelemällä luokka- tai rajapintatyyppi (upper bound), jonka kanssa tyyppiparametrin kiinnittävän arvon tulee olla yhteensopiva.
- Rajoitus ilmaistaan avainsanalla **extends** myös, kun raja on rajapintatyyppiä.
- // Laatikkoon kelpaavat vain nisäkkäät ja jälkeläiset.
  public class Laatikko<T **extends** Nisakas> {
- // Rajoituksen seurauksena Object-tyyppi ei käy
  // tyyppiparametrin arvoksi ja alla oleva ei käänny.
  Laatikko<Object> laatikko = **new** Laatikko<Object>(ville);

Geneerinen tyyppi käännöksessä

- Kääntäjä poistaa (type erasure) geneeriset tyyppiparametrit ja korvaa ne *Object*-tyypillä, ellei tyyppiparametria ole rajoitettu, jolloin korvaava tyyppi on rajan tyyppi.
- Kääntäjä lisää ohjelmaan tarvittavat tyyppimuunnokset.
- **public class** Laatikko<T extends Nisakas> { ...
  private T sisältö;
  public Laatikko(T uusiSisältö) ... { ...
- // Kääntäjän korvaa tyyppiparametrin T tyypillä Nisakas.
  public class Laatikko {
  private Nisakas sisältö;
  public Laatikko(Nisakas uusiSisältö) ... { ...

Raakatyyppi

- Java kiinnittää automaattisesti tyyppiparametrin arvoksi *Object*-tyypin, kun geneeristä tyyppiä käytetään ilman, että ohjelmoija itse määrittelee arvon.
  - Javan automaattisesti antamaa tyyppiä kutsutaan raa'aksi (raw type).
- // Luodaan laatikko raa'asti tyyppiä kiinnittämättä.
  // ja asetetaan Ville laatikkoonsa.
  Laatikko laatikko = **new** Laatikko(ville);
- Kääntäjä varoittaa raa'asta tyypistä:

  Note: LaatikkoTesti.java uses unchecked or unsafe operations.

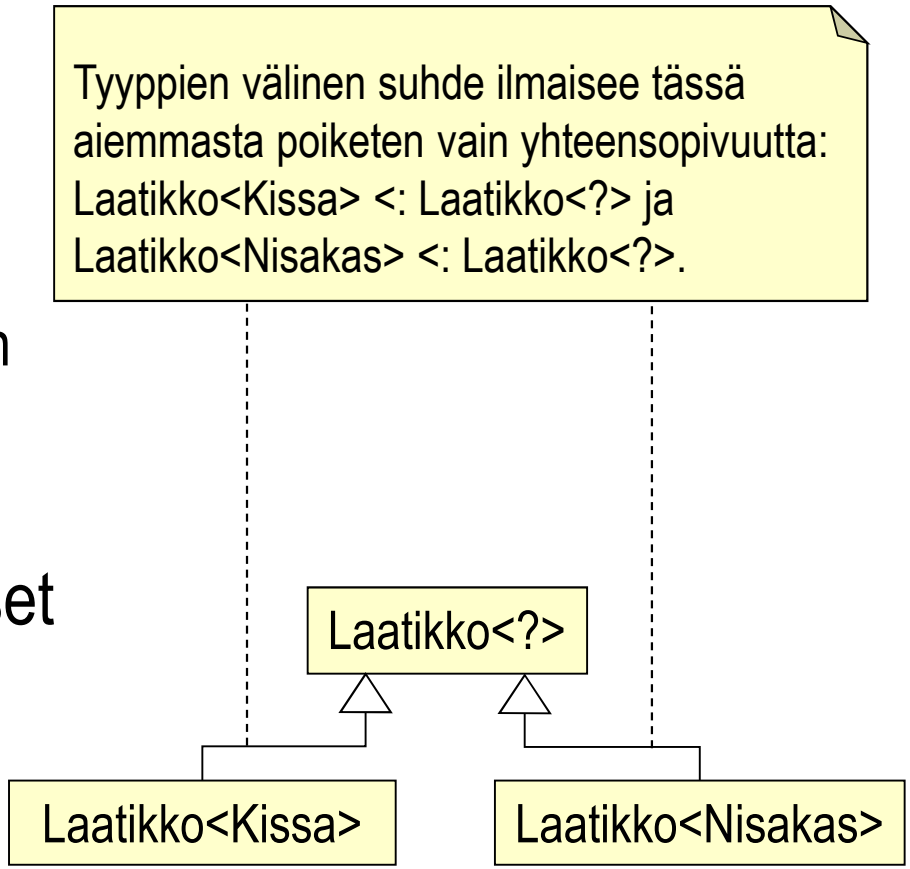
  Note: Recompile with -Xlint:unchecked for details.

Geneeristen tyyppien yhteensopivuus

- Geneeriset tyypit eivät ole yhteensopivia (eivät periydy toisistaan), vaikka tyyppiparametrien arvot ovat yhteensopivia (periytymissuhteessa).
  - Esimerkiksi luokka *Laatikko*<*Kissa*> ei sovi yhteen luokan *Laatikko*<*Nisakas*> kanssa, vaikka *Kissa* <: *Nisakas*.
  - Molemmat luokat periytyvät *Object*-luokasta.
- Geneerinen yhteensopivuus ilmaistaan kysymysmerkillä, joka merkitsee tyypin olevan tuntematon (“villi kortti”).
  - Yhteensopivuus voidaan määrittää parametrille, attribuutille, muuttujalle ja paluuarvolle.
  - Yhteensopivuuden ilmaus on eräs tyyppiparametrin arvo.

Geneeristen tyyppien yhteensopivuus

- Pelkällä tuntemattomalla tyypillä määritelty yhteensopivuus on laajin mahdollinen.
  - Käytetään, jos *Object*-tyypin piirteet riittävät tai metodi ei riipu tyyppiparametrista.
- Esimerkiksi kaikki geneeriset laatikkotyypit ovat yhteensopivia tuntemattomalla arvolla annetun tyypin *Laatikko*<*?*> kanssa.



Geneeristen tyyppien yhteensopivuus

```
public class LaatikkoTesti {
   // Tulostaa minkä tahansa geneerisen laatikon tiedot.
   private static void tulosta(Laatikko<?> laatikko) {
      Object sisällä = laatikko.sisältö();
      System.out.println(sisällä);
   }
   public static void main(String[] args) {
      ...
      // Luodaan laatikkoja erilaisilla tyyppiparametrin arvoilla.
      Laatikko<Kissa> kissanLaatikko = new Laatikko<Kissa>(ville);
      Laatikko<Ihminen> ihmisenLaatikko = new Laatikko<Ihminen>(arska);

      // Tulostetaan laatikkojen sisällöt.
      tulosta(kissanLaatikko);
      tulosta(ihmisenLaatikko);
   }
}
```

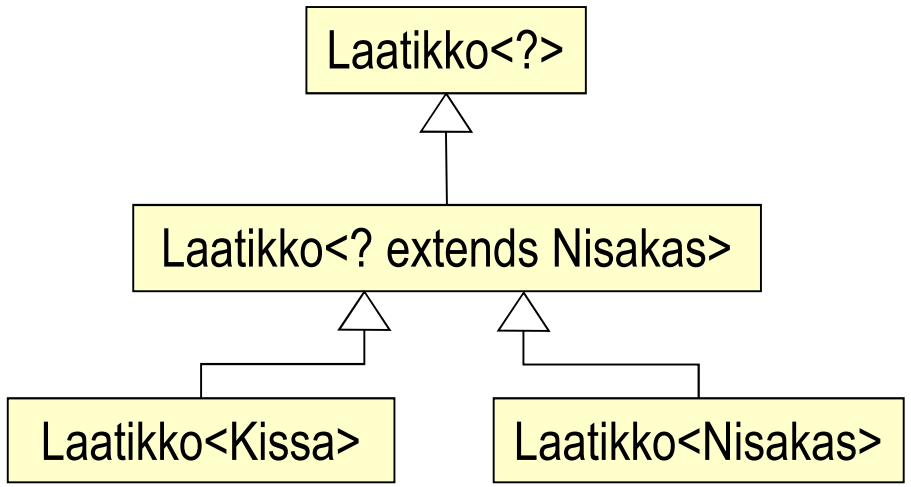
Geneeristen tyyppien yhteensopivuus

- Yhteensopivuutta voidaan rajoittaa antamalla yhdessä kysymysmerkin ja avainsanan kanssa luokka- tai rajapintatyyppi *B*, joka on tyyppiparametrin kiinnittävän arvon yli- tai alityyppi *X*.
  - Avainsana **extends** ilmaisee, että tyyppiparametrin arvon on oltava rajatyypin tyyppiä tai sen alityyppiä (*X* <: *B*), jotta tyypit sopivat yhteen.
  - Avainsana **super** ilmaisee, että tyyppiparametrin arvon on oltava rajatyypin tyyppiä tai sen ylityyppiä (*B* <: *X*), jotta tyypit sopivat yhteen.

Geneeristen tyyppien yhteensopivuus

- Kaikki *Nisakas*-tyyppiin tai sen alityyppiin kiinnitetyt laatikkotyypit ovat yhteensopivia tyypin *Laatikko<? extends Nisakas>* kanssa, joka puolestaan sopii yhteen tyypin *Laatikko<?>* kanssa.

Tyyppien välinen suhde ilmaisee tässä vain yhteensopivuutta:
- Laatikko<Kissa> <: Laatikko<? extends Nisakas>
- Laatikko<Nisakas> <: Laatikko<? extends Nisakas>
- Laatikko<? extends Nisakas> <: Laatikko<?>



Geneeristen tyyppien yhteensopivuus

```
public class LaatikkoTesti {
   // Tulostaa minkä tahansa nisäkkään sisältävän laatikon tiedot.
   private static void tulosta(Laatikko<? extends Nisakas> laatikko) {
      Nisakas sisällä = laatikko.sisältö();
      System.out.println(sisällä);
   }
   public static void main(String[] args) {
      ...
      // Luodaan laatikkoja erilaisilla tyyppiparametrin arvoilla.
      Laatikko<Kissa> kissanLaatikko = new Laatikko<Kissa>(ville);
      Laatikko<Ihminen> ihmisenLaatikko = new Laatikko<Ihminen>(arska);

      // Tulostetaan laatikkojen sisällöt.
      tulosta(kissanLaatikko);
      tulosta(ihmisenLaatikko);
   }
}
```

Geneeriset tyypit ja periytyminen

- Tosin kuin “tavallisten” viitetyyppien yhteydessä tyyppien geneerisestä yhteensopivuudesta ei seuraa tyyppien periytymissuhdetta.
- Periytymissuhde määritellään tuttuun tapaan **extends**-avainsanalla.
- Periytymisen määrittelyssä aliluokalle pitää antaa sama tyyppiparametri, jota käytetään yliluokassa.
  - Aliluokassa voidaan esitellä omia tyyppiparametreja.
  - Periytymisestä seuraa aiemmin opittuun tapaan yhteensopivuus.
- // Taikalaatikko<T, U> <: Laatikko<T>
  public class Taikalaatikko<T, U> **extends** Laatikko<T> ...