Föreläsning 17

Elias Castegren

Inkapsling

Uppföljningsmötet på fredag vigs åt de som inte är klara med Z101 (se Piazza)

Kolla in Java Visualizer!

http://cscircles.cemc.uwaterloo.ca/java_visualize

Var är vi i schemat?

Typ	Innehåll
F15	Introduktion till objektorientering. Från C till Java. Skräpsamling 1/2.
	Huvudmål: A2, M37, N40
F16	Skillnad klass och objekt. Konstruktorer. Instantiering, referenser, klassvariabler. Identitet och ekvivalens.
	Huvudmål: H19, H20
F17	Inkapsling. Inre och nästlade klasser. Wrapperklasser.
	Huvudmål: A2, G15, G16, G17
F18	Arv och klasshierarkier. Överlagring. Overriding. Överlagring av konstruktorer. Separation of Concerns.
	Huvudmål: B4, B5, B6, K30, K31, K32

- Skillnad klass och objekt

  ”En klass är en mall för hur en viss sorts objekt skapas”

- Konstruktorer och instantiering

  Varje klass har en eller flera konstruktorer som anropas när objektet skapas

- Referenser

  Inga pekare. Alla objekt nås genom referenser.

- Identitet och ekvivalens

  ”Samma objekt” vs. ”Likadana objekt”. `x == y` vs. `x.equals(y)`

Klassvariabler

- Ingenting existerar utanför klasserna
- Metoder och attribut hör till specifika objekt

  `x.getFoo()` betyder något annat än `y.getFoo()`

- Kan vi ha globala variabler?
- Hur kör `main`-metoden om det inte finns något objekt när programmet startar!?

```
public class Foo {
    public static void main(String args[]) {
        ...
    }
}
```

Klassvariabler

- Metoder och attribut som markeras som `static` hör till *klassen*
- Saknar mottagare

  jämför `x.getFoo()` — Skicka meddelandet `getFoo` till `x`.

  med `Foo.bar()` — Anropa `bar` i klassen `Foo`.

- Statiska metoder (klassmetoder) är som funktioner.
- Statiska attribut (klassvariabler) är som globala variabler.

- Statisk kod kan inte anropa icke-statisk kod!

  Varför?

Var är vi i schemat?

Typ	Innehåll
F15	Introduktion till objektorientering. Från C till Java. Skräpsamling 1/2.
	Huvudmål: A2, M37, N40
F16	Skillnad klass och objekt. Konstruktorer. Instantiering, referenser, klassvariabler. Identitet och ekvivalens.
	Huvudmål: H19, H20
F17	Bli ännu lite bättre på att förstå Java-kod
	Huvudmål: A2, G15, G16, G17
F18	Arv och klasshierarkier. Överlagring. Overriding. Överlagring av konstruktorer. Separation of Concerns.
	Huvudmål: B4, B5, B6, K30, K31, K32

- **Arv (mycket kort)**

  En klass kan *ärva* av en annan klass

  Klassen A ärver av Klassen B ⇒ En A är (också) en B

  En pudel är (också) en hund. En hund är (också) ett djur.

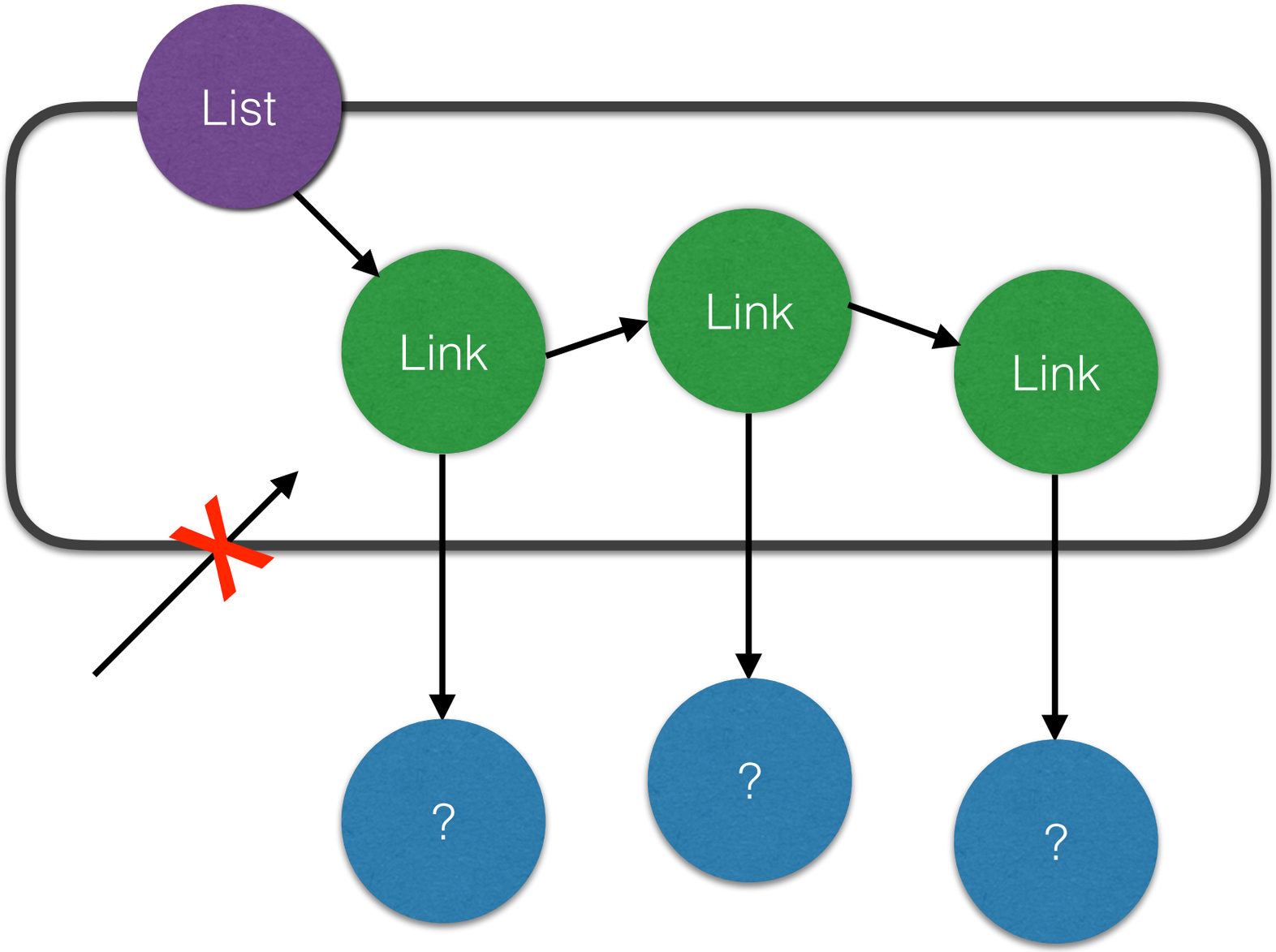
```
public class Poodle extends Dog {
    …
}
```

```
public class Dog extends Animal {
    …
}
```

  Alla klasser ärver av klassen Object (ett objekt är (också) ett Object).

Inkapsling

- Relaterat till modularisering och informationsgömning.
- Inkapsling avser *ägarskap* av vissa objekt:



Inkapsling

- ”I Java får man inkapsling om man använder `private`!”

Inte sant!

```
public class Customer {
  private Account bankAccount;

  public Account getAccount() {
    return this.bankAccount;
  }

  public void stealAccount(Customer c) {
    this.bankAccount = c.bankAccount;
  }
}
```

Läckage!

Läckage!

- `private` avser synlighet *per klass*, inte per objekt!
- `private` **kan** användas för att uttrycka att **avsikten** är inkapsling (men inga garantier ges)

Inkapsling

- Ett sätt att undvika läckage är att kopiera alla (ägda) objekt som passerar gränssnittet:

```
public class Customer {
  private Account bankAccount;

  public Account getAccount() {
    Account accountCopy = this.bankAccount.clone();
    return accountCopy;
  }
```

```
public class Theif {
  public void steal(Customer c) {
    Account account = c.getAccount();
    account.setBalance(0);
  }
```

Ändrar en kopia!

Inkapsling

- Ett sätt att undvika läckage är att kopiera alla (ägda) objekt som passerar gränssnittet:

```
public class Customer {
  private Account bankAccount;

  public Customer(Account account) {
    this.bankAccount = account;
  }

  public Account getAccount() {
    Account accountCopy = this.bankAccount.clone();
    return accountCopy;
  }
```

Vad är fel här?

```
public class Theif {
  public void trick(Customer p) {
    Account account = new Account(10000)
    Customer c = new Customer(account)
    account.setBalance(0);
  }
```

Ändrar kundens konto!

Inkapsling

- Ett sätt att undvika läckage är att kopiera alla (ägda) objekt som passerar gränssnittet:

```
public class Customer {
  private Account bankAccount;

  public Customer(Account account) {
    this.bankAccount = account.clone();
  }

  public Account getAccount() {
    Account accountCopy = this.bankAccount.clone();
    return accountCopy;
  }
```

```
public class Theif {
  public void trick(Customer p) {
    Account account = new Account(10000)
    Customer c = new Customer(account)
    account.setBalance(0);
  }
```

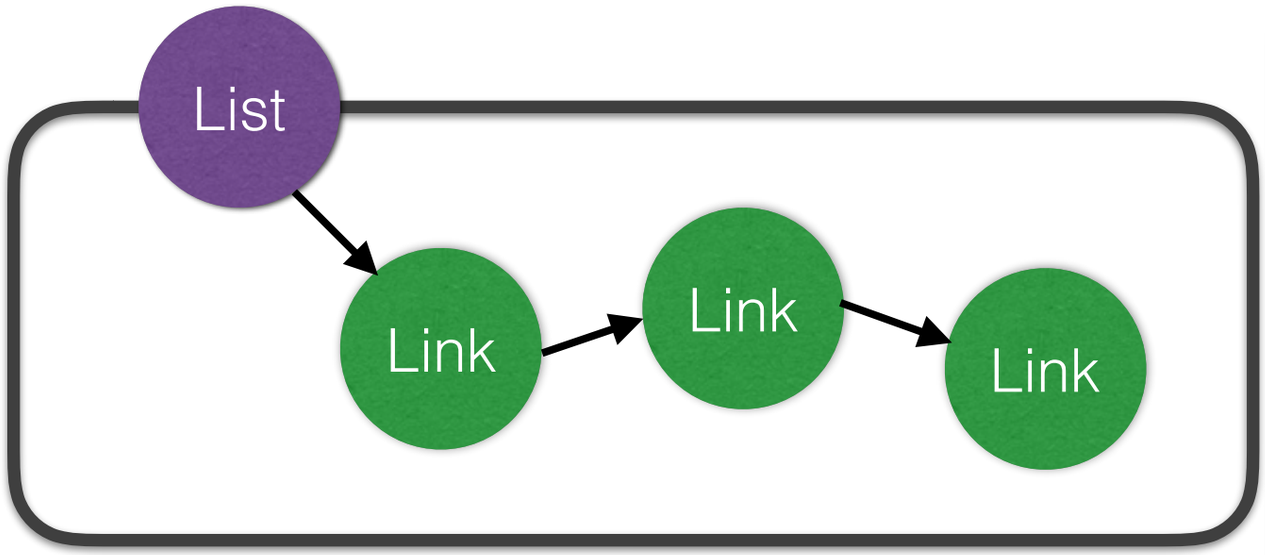
Ändrar den lokala kopian!

Inkapsling (Nästlade klasser)

- Inkapsling sker på två nivåer:

  Objektnivå — Listans länkar (objekten) är inkapslade i Listan (objektet)

  Klassnivå — Klassen `Link` är bara meningsfull för klassen `List`




```
public class List {
  private class Link {
    private Object elem;
    private Link next;
    ...
  }
  private Link first;
  ...
}
```