TDDC77
Objektorienterad Programmering

Föreläsning 9

Sahand Sadjadee
IDA, Linköpings Universitet

Outline

- Projektgenomgång
- Tentagenomgång
- Hierarkier
- Polymorfism(17.3)
- Nyckelordet `instanceof` (17.2)
- Metoden `equals()` (`14.5.3`)
- Generics

Projektgenomgång

LINKÖPINGS UNIVERSITET

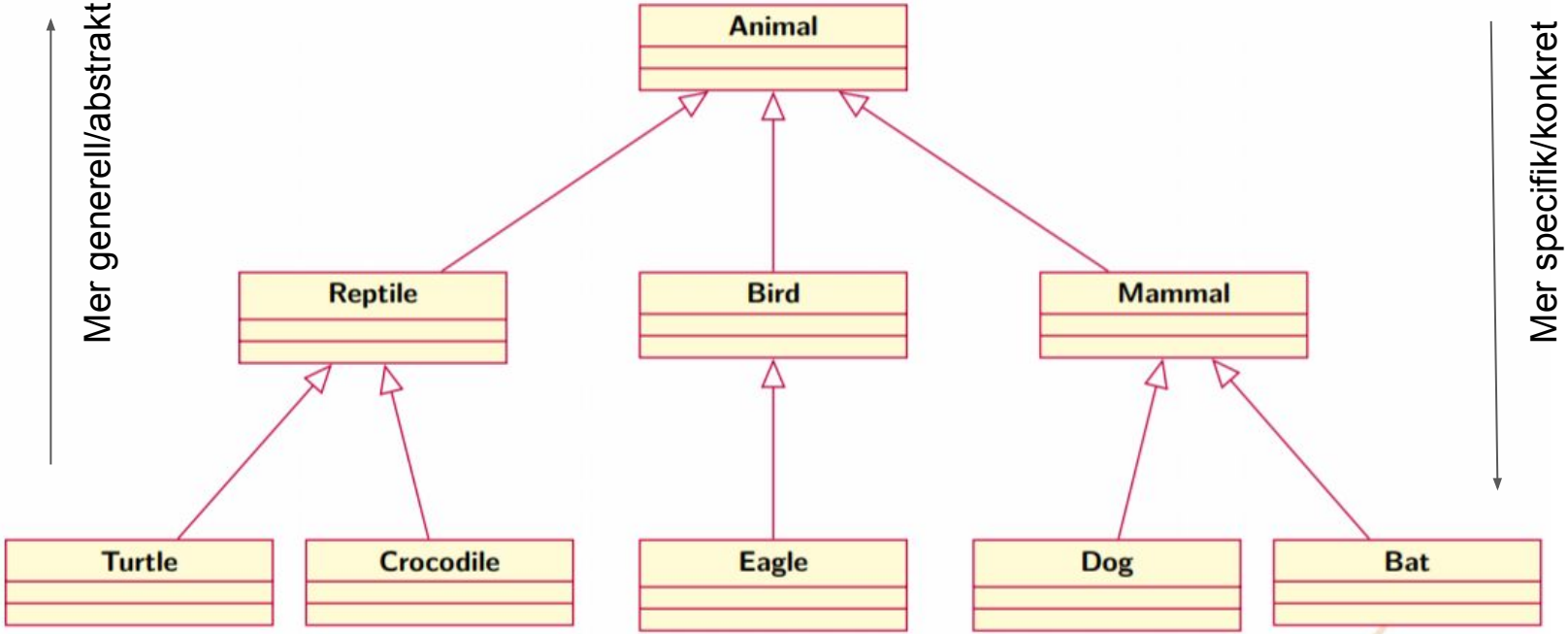
Tentagenomgång

LINKÖPINGS UNIVERSITET

Hierarkier

Klasshierarki

“A hierarchy is an arrangement of items in which the items are represented as being "above", "below", or "at the same level as" one another. Hierarchy is an important concept in a wide variety of fields, such as philosophy, mathematics, computer science, organizational theory, systems theory, and the social sciences.”



LINKÖPINGS UNIVERSITET

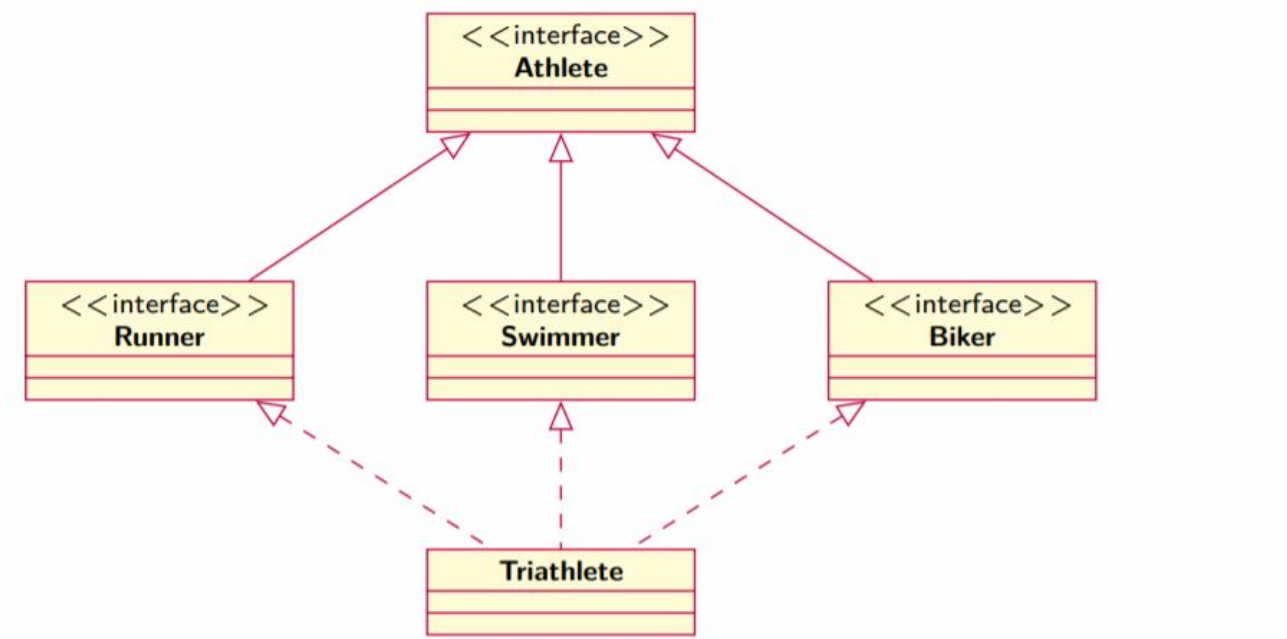
Interface

- Nyckelordet `interface` används för att deklarera interface.
- Interface är klasser som bara innehåller publika abstrakta metoder och publika statiska konstanter.
- Alla metoder i interface är automatiskt publika och abstrakta.
- Alla konstanter i interface är automatiskt publika och statiska.
- En klass kan implementera/ärva från **flera** interface.
- Nyckelordet `imeplements` används i klassdeklarationen för att implementerea interface.

Interface

```
public interface Transform {
    void transform();
}

public interface Fly {
    void fly();
}

public class Car implements Fly, Transform {

    @Override
    public void fly() {
        System.out.println("I can Fly!!");
    }

    @Override
    public void transform() {
        System.out.println("I can Transform!!");
    }
}
```

Klasshierarki




```
class Triathlete implements Runner, Swimmer, Biker {
   ...
}
```

Klasshierarki

	Class	**Interface**
Class	ärver	implementerar
Interface	-	ärver

Polymorphism

Polymorfism

- Polymorfism innebär inom objektorienterad programmering att flera olika subklasser under en superklass kan hanteras som om de vore instanser av superklassen.
- Det innebär att klasser med olika behov vad gäller implementering av en viss metod, ändå kan anropas på samma sätt.
- Den verkställande programkoden finns i respektive subklass, medan det gemensamma gränssnittet definieras i superklassen.

Polymorfism

```
public class Polymorphism{
    public static void Main(String[] args){
        Animal[] animalsInTheGarden = new Animal[2];
        animalsInTheGarden[0] = new Cat("Meo");  //is-a relation mellan Animal och Cat.
        animalsInTheGarden[1] = new Dog("Goldy"); //is-a relation mellan Animal och Dog.


        for (Animal t: animalsInTheGarden){
            t.says();
        }
    }
}
```

instanceof

- Är ett nyckelord som används för att utreda om objektet är instans av en viss klass eller inte.

```
class Simple1{
      public static void main(String args[]){
            Simple1 s=new Simple1();
            System.out.println(s instanceof Simple1);//true
      }
}
```

equals()

- Metoden tillhör till `java.lang.Object`
- Används för att jämföra objekt.
- Default-implementationen som ligger klassen Object jämför bara referenserna och inte innehållen.
- Ska override:as för att kunna jämföra innehållen.

```
public boolean equals(Object o) {
    if (o == this)
        return true;
    if (!(o instanceof Money))
        return false;
    Money other = (Money)o;
    boolean currencyCodeEquals = (this.currencyCode == null && other.currencyCode == null)
      || (this.currencyCode != null && this.currencyCode.equals(other.currencyCode));
    return this.amount == other.amount && currencyCodeEquals;
}
```

Generics

Generics

En generictyp är en klass eller ett interface som är parametriserad med en eller flera typer.

```
class MyPair<T1, T2> {
   private T1 first;
   private T2 second;

   MyPair(T1 t1, T2 t2){
     first = t1;
     second = t2;
   }

   public String toString(){
     return "(" + first  + ", " + second + ")";
   }
 }
```

```
class MyPairDriver {
   public static void main(String[] args) {

    MyPair<String,Integer> p1 = new MyPair<String,Integer>("Monday",1);
    System.out.println(p1);

    MyPair<Integer,Integer> p2 = new MyPair<Integer,Integer>(3,7);
    System.out.println(p2);
   }
}
```

LINKÖPINGS UNIVERSITET

Thanks for listening!

LINKÖPINGS UNIVERSITET