

EDA016 Programmeringsteknik för D

Läsvecka 7: Arv

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2015

7 Arv

- Att göra denna vecka
- Grundläggande terminologi för arv
- Klassificering och abstrakta klasser
- Operatörn `instanceof`
- Klassen `Object`
- `Super` i konstruktor och `protected`

Att göra i Vecka 7: Förstå arv. Repetera inför kontrollskrivning.

- 1 Läs följande kapitel i kursboken:
9.1, 9.3, 9.7-9.9, 11.2, 12.6
Begrepp: arv, subclass, superklass, **extends**, **abstract**,
operatör **instanceof**, Object, **super**, equals,
- 2 Gör övning 7: registrering
- 3 Träffas i samlarbetsgrupper och hjälp varandra
- 4 Gör Lab 6: Implementera Turtle och ColorTurtle.
- 5 Plugga inför **obligatorisk** kontrollskrivning.

Vad är arv? Motivering och terminologi

- Med hjälp av **arv** mellan klasser kan man göra så att en klass **ärver** ("får med sig") innehållet i en *annan* klass.
- Varför vill man det?
 - 1 Dela upp ansvar mellan klasser och bryta ut gemensamma delar så att man slipper duplicerad kod.
 - 2 Skapa en klassificering av objekt utifrån relationen **X är en Y**.
Exempel: En gurka är en grönsak. En cykel är ett fordon.
- Nyckelordet **extends** används för att ange arv i Java.
Exempel: **class** TalkingRobot **extends** Robot
- Klassen som ärver (utökar) kallas **subklass**
- Klassen som blir utökad kallas **superklass** (även *basklass*)
- Läs mer om arv (eng. *inheritance*) här:
https://sv.wikipedia.org/wiki/Arv_%28programming%29

Exempel: Robot

Krav:

Det ska finnas två sorters robotar: Robot och TalkingRobot. Båda kan arbeta, men bara den senare kan prata.

En möjlig **design** om vi *inte* använder arv:

Robot

```
/** Arbeta */  
public void work();
```

TalkingRobot

```
/** Arbeta */  
public void work();  
  
/** Prata */  
public void talk();
```

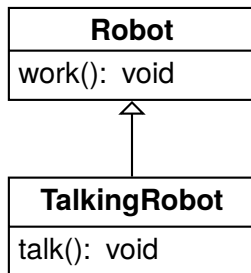
Utökning med copy-paste ger duplicerad kod

```
1 package week07.robotduplicate;
2
3 class Robot {
4     public void work() {
5         System.out.println("Robot is workign.");
6     }
7 }
8
9 class TalkingRobot {
10     public void work() { // Code duplication with copy-paste.
11         System.out.println("Robot is workign."); // Need to fix misspelling in 2 places :(
12     }
13
14     public void talk() {
15         System.out.println("I shall not harm humans.");
16     }
17 }
18
19 public class RobotNoInheritance {
20     public static void main(String[] args) {
21         Robot wallE = new Robot();
22         wallE.work();
23         TalkingRobot c3po = new TalkingRobot();
24         c3po.talk();
25         c3po.work();
26     }
27 }
```

Utökning med arv – vi slipper duplicera kod

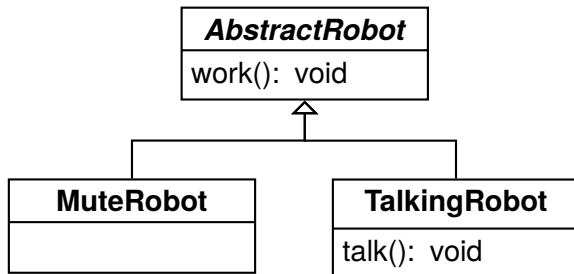
```
1 package week07.robotinherit;  
2  
3 class Robot {  
4     public void work() {  
5         System.out.println("Robot is working.");  
6     }  
7 }  
8  
9 class TalkingRobot extends Robot { // TalkingRobot inherits the contents of Robot  
10     public void talk() {  
11         System.out.println("I shall not harm humans.");  
12     }  
13 }  
14  
15 public class RobotInheritance {  
16     public static void main(String[] args) {  
17         Robot walle = new Robot();  
18         walle.work();  
19         TalkingRobot c3po = new TalkingRobot();  
20         c3po.talk();  
21         c3po.work();  
22     }  
23 }
```

Illustrera arvsrelationer i diagram

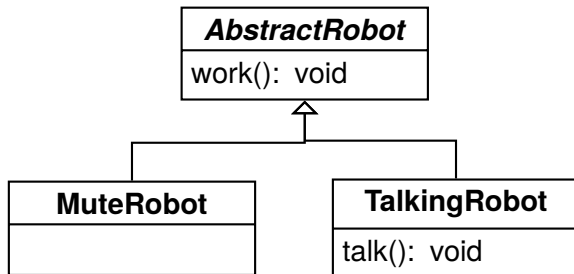


Class Diagram, Unified Modelling Language (UML), Ankboken sid 147.

Klassificering av robotar



Klassificering av robotar



- **Abstrakta klasser** används som bas för klassificering
- Deklaration i Java: **abstract class** ClassName
- Försök att instantiera abstrakta klasser ger kompileringsfel.
- MuteRobot är en **konkret klass** som kan instantieras och ärver `work()`-metoden från **AbstractRobot**

Exempel: klassificeringsmodell i Java

```
1 package week07.robotclassification;
2
3 public abstract class AbstractRobot {
4     public void work() {
5         System.out.println("Robot is working.");
6     }
7 }
```

```
1 package week07.robotclassification;
2
3 public class TalkingRobot extends AbstractRobot {
4     public void talk() {
5         System.out.println("I shall not harm humans.");
6     }
7 }
```

```
1 package week07.robotclassification;
2
3 public class MuteRobot extends AbstractRobot {
4     //empty class body; nothing added to AbstractRobot
5 }
```

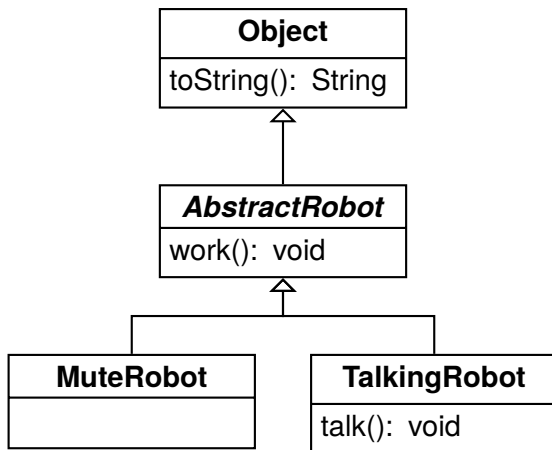
Exempel: test av klassificeringsmodell

```
1 package week07.robotclassification;
2
3 public class TestRobots {
4     public static void main(String[] args) {
5         MuteRobot wallE = new MuteRobot();
6         TalkingRobot c3po = new TalkingRobot();
7         // AbstractRobot r = new AbstractRobot(); //compile error: "Cannot instantiate..."
8         wallE.work();
9         // wallE.talk(); //compile error: "The method talk() is undefined..."
10        c3po.talk();
11        c3po.work();
12    }
13 }
```

Operatör instanceof

```
1 package week07.robotclassification;
2
3 public class TestInstanceOf {
4     public static void main(String[] args) {
5         MuteRobot wallE = new MuteRobot();
6         TalkingRobot c3po = new TalkingRobot();
7         if (wallE instanceof MuteRobot) {
8             System.out.println("Wall-E kan inte prata");
9         }
10        //if (wallE instanceof TalkingRobot) { // compile error
11        //    System.out.println("Wall-E kan inte prata");
12        //}
13        if (c3po instanceof TalkingRobot) {
14            System.out.println("C3PO kan prata");
15        }
16        if (wallE instanceof AbstractRobot){
17            System.out.println("Wall-E kan arbeta");
18        }
19        if (c3po instanceof AbstractRobot){
20            System.out.println("C3PO kan arbeta");
21        }
22        if (wallE instanceof Object && c3po instanceof Object){
23            System.out.println("Wall-E och C3PO är javaobjekt.");
24        }
25    }
26 }
```

Alla objekt är implicita instanser av klassen Object



Super i konstruktör och protected

```
1 package week07.superconstructor;  
2  
3 public class Robot {  
4     protected String name;  
5  
6     public Robot(String name) {  
7         this.name = name;  
8     }  
9  
10    public void work() {  
11        System.out.println(name + " is working.");  
12    }  
13 }
```

```
1 package week07.superconstructor;  
2  
3 public class TalkingRobot extends Robot {  
4     public TalkingRobot(String name) {  
5         super(name);  
6     }  
7  
8     public void talk() {  
9         System.out.println(name + " shall not harm humans.");  
10    }  
11 }
```