Arv

- En klass som ärver är en **subklass** till den klass den ärver ifrån
- Den klass subklassen ärver ifrån kallas **superklass**
- Alla klasser är subklasser utom **Object**
- Nyckelordet ***extends*** används för att visa att en klass ärver från en annan

```
public class Car extends Vehicle
{

}
```

Klasser som ärver

- Ärver funktionalitet och egenskaper från sin superklass

```
//Superclass
public class Vehicle
{
        private int speed = 0;

        public void accelerate ()
        {
            speed++;
        }
}

//Subclass
public class Car extends Vehicle
{
}

Car car = new Car();
car.accelrate();
```

← Metod som finns i superklassen ärvs till subklassen

Konstruktorer och arv

- När en subklass instansieras *måste* superklassens konstruktor anropas
- Nyckelordet ***super*** används för att anropa superklassens konstruktor
- Detta måste vara det första som sker i subklassens konstruktor
- Om super utelämnas infogar kompilatorn det automatisk
  Notera: Detta gäller bara om superklassen har en **default konstruktor**

```
//Subclass
public class Car extends Vehicle
{
    public Car()
    {
        super(); ←Här anropas superklassens konstruktor
    }
}
```

Konstruktorer och arv fortsättning

- Exempel på då **vi** måste anropa superklassens konstruktor:

```
//Superclass
public class Vehicle
{
        private int speed;

        public Vehicle(int initialSpeed)
        {
                speed = initialSpeed;
        }

}

//Subclass
public class Car extends Vehicle
{
        public Car()
        {
                //super(); ← Detta går inte. Superklassen har ingen default konstruktor
                super(0); ← Anropar superklassens konstruktor
        }
}
```

Åsidosätta metoder – eng. *Override*

- När en subklass definierar en metod som är *exakt* som en metod i dess superklass

```
//Superclass
public class Vehicle
{
        public void accelerate()
        {
        }
}

//Subclass
public class Car extends Vehicle
{
        @Override
        public void accelerate() ← Detta  ”skymmer” metoden i superklassen
        {
        }
}

Car car = new Car();
car.accelerate(); ←Anropar metoden accelerate i Car-klassen
```

Komma åt metoder i superklassen

- Nyckelordet ***super*** används också för att komma åt metoder i en superklass

```
//Superclass
public class Vehicle
{
        private int speed;

        public void accelerate()
        {
            speed++;
        }
}

//Subclass
public class Car extends Vehicle
{
        @Override
        public void accelerate()
        {
            if(started)
            {
                super.accelerate(); ← Anropar superklassens metod
            }
        }
}
```

final

- **final** används för att omöjliggöra arv

  //Superclass
  public final class Vehicle
  {
  }

  //Subclass
  public class Car extends Vehicle ← **Går inte att göra**
  {
  }

  Detta fungerar **INTE** eftersom **Vehicle** är **final** och därmed förbjuder att någon ärver från den

Polymorfism

- Betyder "många former"
- Ett objekt av en subklass kan refereras av en objekts-ref.variabel av superklasstyp

```
//superclass
public class SuperClass()
{
}

//subklass
public class SubClass() extends SuperClass
{
}
```

SuperClass sub = new SubClass(); ← Lagras i en objekts-ref.variabel av superklasstyp

Notera: variabeln **sub** kommer bara åt metoder som finns i SuperClass och dess superklasser - den kommer **inte** år metoder i SubClass fastän det är ett objekt av den typen som skapats

Hur uppnås polymorfism?

- Metodöverlagring - ett metodnamn flera argumentlistor
- Metodåsidosättande (override) genom subklassning - en metod flera implementationer
- Metodåsidosättande (override) genom interface-implementation - en metod flera implementationer

Nyttan med Polymorfism

- Tycks ge superklassen många beteenden genom sina subklasser
- Vilket objekt som refereras av en objekts-ref.variabel behöver inte vara känt under kompilering
- Ger stor flexibilitet

Hur fungerar det?

- Vilken metod som anropas avgörs **INTE** av vilken objekts-ref.variabel det är utan av vilket objekt det är som den refererar till

```
//superclass
public class SuperClass()
{
      public void printMessage()
      {
            System.out.println("SuperClass");
      }
 }
```

-

```
//subklass
public class SubClass() extends SuperClass
{
      public void printMessage()
      {
            System.out.println("SubClass");
       }
 }
```

SuperClass sub = new SubClass(); ← Lagras i en objekts-ref.variabel av superklasstyp
sub.printMessage(); ← Skiver ut "SubClass"

Anropar printMessage i **SubClass** fastän det är en objekts-ref.variabel av Superklasstyp

Varför fungerar det?

- Metoden som exekveras är åsidosatt (override)
- Vilken metod som anropas avgörs först under **runtime** INTE under compile-time
- Eftersom objekts-ref.variabeln refererar ett objekt av subklasstyp som åsidosätter metoden i superklassen exekveras den - den är den första metod som hittas