EDAA20
Programmering och databaser

Föreläsning 11 – Listor och klassen `ArrayList`

2023-10-02, Niklas Fors

Från **Kursplanen** (se Moodle-sida)

Programmering

	Läsvecka 1	Läsvecka 2	Läsvecka 3	Läsvecka 4	Läsvecka 5	Läsvecka 6	Läsvecka 7	
Mån	F1	F3	F5	F7	F9	F11	F13	
Tis	F2	F4	F6	F8	F10	F12	F14	
Ons	L1	L3	L4	L6	L8	L	L11	
Fre	L2	L	L5	L7	L9	L10	L	Instuderings-vecka
Ons/Tors	R	R	R	R	R	R	R	

F föreläsningar, L datorlaborationer (obligatoriska), R resurstid

Laboration 9	matriser, Memory-spel . . . . .
Laboration 10	använda klassen ArrayList, spelkort

Tenta, måndag 23 oktober

Listor

En ***lista*** är en följd element av en viss typ

Klassen `ArrayList` används normalt för listor,
där element enkelt kan läggas in och tas bort

Är listor bättre än vektorer?

Nackdelar med vektorer

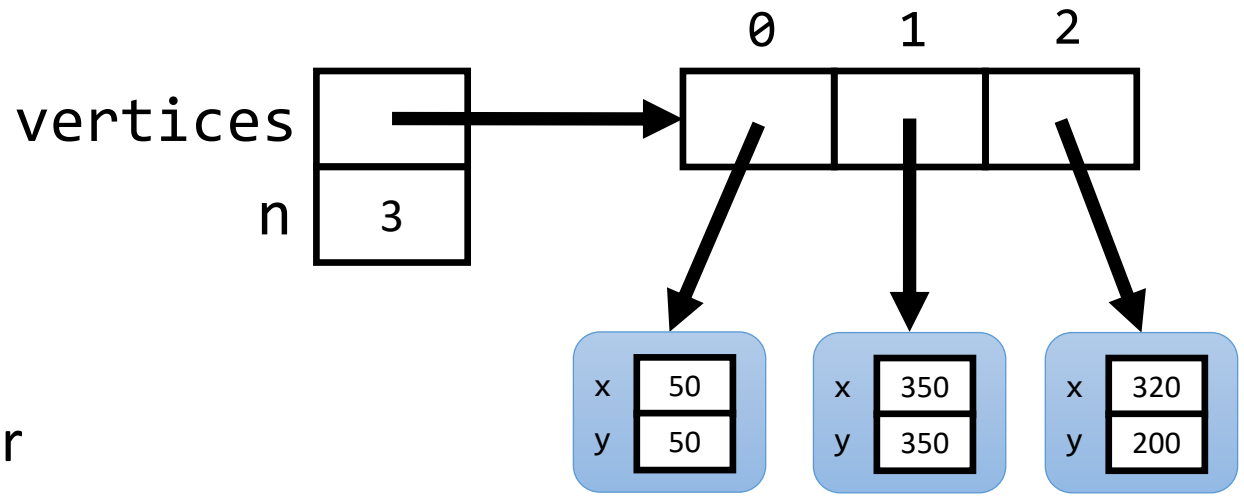
När man skapar vektorer måste storleken anges:

```
int[] v = new int[5];
Point[] vertices = new Point[10];
```

Hur ska vi hantera om

- vektorn är full?
- element sätts in i mitten?
- element tas bort?

Vektorn blir full

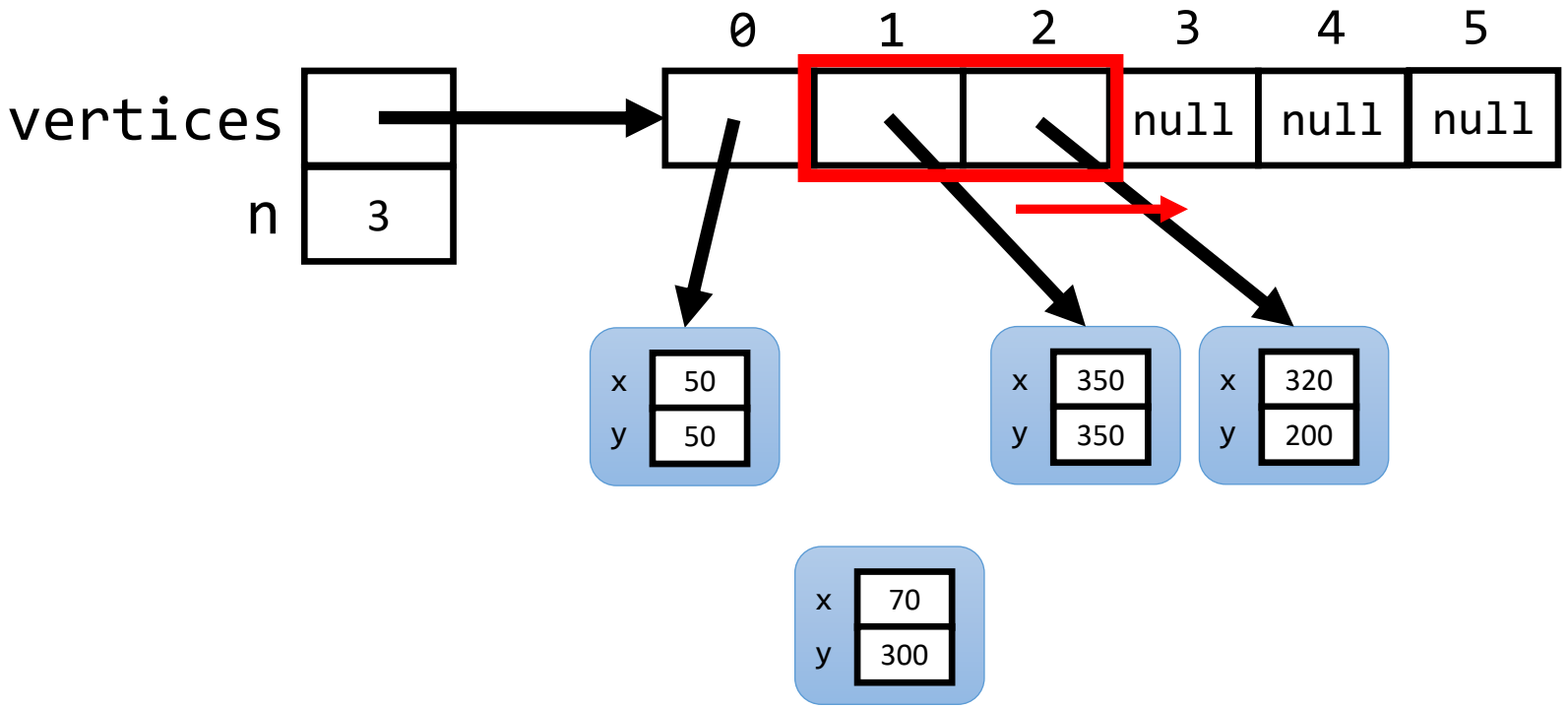


Om vektorn blir full får man:

1. Skapa en ny större vektor
2. Kopiera alla elementen till ny vektor
3. Låt `vertices` referera till ny vektor

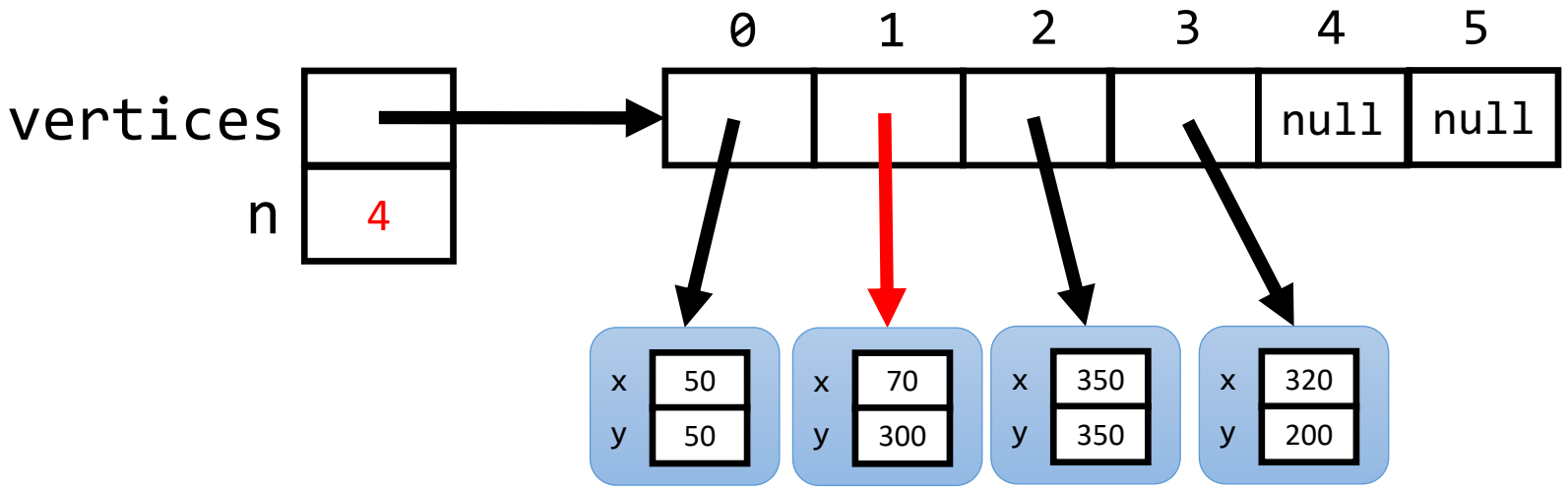
Lägga till element i mitten

Om man vill lägga in punkten (70, 300) på index 1 får man flytta punkterna på index 1-2 ett steg till höger.



Lägga till element i mitten

Om man vill lägga in punkten (70, 300) på index 1 får man flytta punkterna på index 1-2 ett steg till höger.



Nackdelar med vektorer

När man skapar en vektor måste man ange storleken

Beroende på tillämpning behöver man själv hantera:

- om vektorn är full
- insättning av element i mitten
- borttagning av element

Listor

Klassen `ArrayList` löser problemen åt oss

(`ArrayList` använder internt en vektor)

Klassen `ArrayList`

Skapa en **tom** lista där punkter kan läggas in:

```
ArrayList<Point> points = new ArrayList<Point>();
```

Klassen `ArrayList`

Skapa en **tom** lista där punkter kan läggas in:

```
ArrayList<Point> points = new ArrayList<Point>();
```

Inom <…> anges vilken sorts objekt som listan innehåller,
dvs, ett klassnamn (Point, String, ...)

Klassen `ArrayList`

Skapa en **tom** lista där punkter kan läggas in:

```
ArrayList<Point> points = new ArrayList<Point>();
```

Lägg till en punkt i slutet av listan:

```
points.add(new Point(50, 50));
```

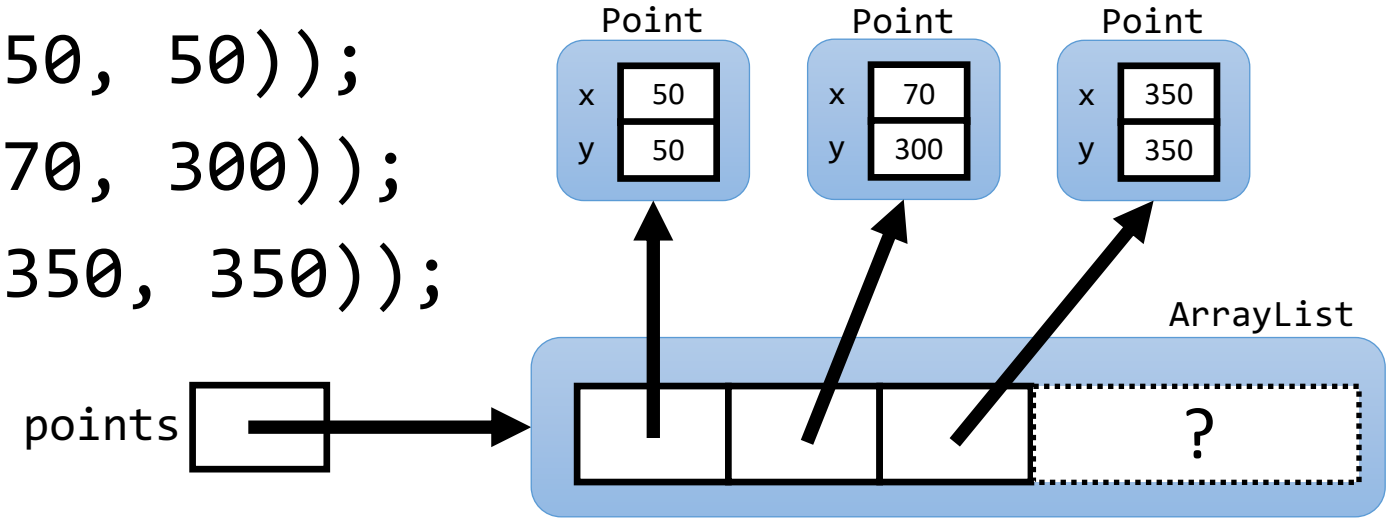
Klassen `ArrayList`

Skapa en **tom** lista där punkter kan läggas in:

```
ArrayList<Point> points = new ArrayList<Point>();
```

Lägg till en punkt i slutet av listan:

```
points.add(new Point(50, 50));
points.add(new Point(70, 300));
points.add(new Point(350, 350));
```



Vektor vs Lista

```java
// Skapa vektor med 2 platser
Point[] points = new Point[2];

// Lägg in punkter på platser
points[0] = new Point(50, 50);
points[1] = new Point(70, 300);

// Hämta första punkten
Point p = points[0];

// Hämta antalet platser
int n = points.length;
```

```java
// Skapa tom lista
ArrayList<Point> points = new ArrayList<Point>();

// Lägg till punkter i slutet av listan
points.add(new Point(50, 50));
points.add(new Point(70, 300));

// Hämta första punkten
Point p = points.get(0);

// Hämta antalet element
int n = points.size();
```

Man anropar metoder på listan

Övning

Vad skriver följande program ut?

```
Point[] pointArray = new Point[4];
ArrayList<Point> pointList = new ArrayList<Point>();

System.out.println(pointArray.length);
System.out.println(pointList.size());
```

Övning

Vad skriver följande program ut?

```
Point[] pointArray = new Point[4];
ArrayList<Point> pointList = new ArrayList<Point>();

System.out.println(pointArray.length);   // 4
System.out.println(pointList.size());    // 0
```

Från början har vektorn en viss storlek och listan är tom

Terminologi: generisk klass

Klassen `ArrayList<E>` är ***generisk***, vilket kräver *typargument* (klassnamn) inom `<...>` vid användning. Typargumentet bestämmer vilka element som kan läggas in i listan.

```
ArrayList<Point> points = new ArrayList<Point>();
points.add(new Point(50, 50));   // OK

points.add("asd");               // Kompileringsfel! Strängen
                                 // "asd" är ingen punkt
```

Viktiga metoder i `ArrayList<E>`

```
/** Skapar en tom lista med element av typen E. */
ArrayList<E>();

/** Returnerar elementet på plats pos. */
E get(int pos);

/** Lägger in obj sist. */
void add(E obj);

/** Tar bort elementet på plats pos, returnerar det borttagna elementet. */
E remove(int pos);

/** Returnerar antalet element. */
int size();
...
```

E ersätts av typargumentet

Till exempel, för `ArrayList<Point>` ersätts E av `Point` i get/add/remove osv.

Standardklasser, import java.util.Classname

List	List<E> är ett gränssnitt som beskriver listor med objekt av parameterklassen E. Man kan lägga in värden av standardtyperna genom att kapsla in dem, till exempel int i Integer-objekt. Gränssnittet implementeras av klasserna ArrayList<E> och LinkedList<E>, som har samma operationer. Man ska inte använda operationerna som har en position som parameter på en LinkedList (i stället en iterator). IndexOutOfBoundsException om någon position är fel. För att operationerna contains, indexOf och remove(Object) ska fungera måste klassen E överskugga funktionen equals(Object). Integer och de andra wrapperklasserna gör det.	
ArrayList	ArrayList<E>();	skapar tom lista
LinkedList	LinkedList<E>();	skapar tom lista
	int size();	antalet element
	boolean isEmpty();	ger true om listan är tom
	E get(int i);	tar reda på elementet på plats i
	int indexOf(Object obj);	index för obj, -1 om inte finns
	boolean contains(Object obj);	ger true om obj finns i listan
	void add(E obj);	lägger in obj sist, efter existerande element
	void add(int i, E obj);	lägger in obj på plats i (efterföljande element flyttas)
	E set(int i, E obj);	ersätter elementet på plats i med obj
	E remove(int i);	tar bort elementet på plats i (efterföljande element flyttas)
	boolean remove(Object obj);	tar bort objektet obj, om det finns
	void clear();	tar bort alla element i listan

Som vanligt kan man se vilka metoder som finns i Snabbreferensen.

Programmet skapar 5 tärningar och kastar dem (som tidigare):

```
Dice[] dices = new Dice[5];

// Skapa tärningarna
for (int i = 0; i < dices.length; i++) {
  dices[i] = new Dice();
}

// Kasta tärningarna
for (int i = 0; i < dices.length; i++) {
   dices[i].roll();
   System.out.println(dices[i].getDots());
}
```

ÖVNING

Skriv färdigt följande program som ska göra samma sak, fast med en lista istället för med en vektor.

```
Dice[] dices = new Dice[5];

// Skapa tärningarna
for (int i = 0; i < dices.length; i++) {
  dices[i] = new Dice();
}

// Kasta tärningarna
for (int i = 0; i < dices.length; i++) {
   dices[i].roll();
   System.out.println(dices[i].getDots());
}
```

```
ArrayList<Dice> dices = new ArrayList<Dice>();

// Skapa tärningarna
for (int i = 0; i < 5; i++) {
  // Skapa tärning och lägg till i listan
}

// Kasta tärningarna
for (int i = 0; i < dices.size(); i++) {
  // Kasta i:e tärningen i listan
  // Skriv ut dess resultat
}
```

Anropa följande metoder på listan:

```
/** Lägger in obj sist. */
void add(E obj);

/** Returnerar elementet på plats pos. */
E get(int pos);
```

ÖVNING

Skriv färdigt följande program som ska göra samma sak, fast med en lista istället för med en vektor.

```
Dice[] dices = new Dice[5];

// Skapa tärningarna
for (int i = 0; i < dices.length; i++) {
  dices[i] = new Dice();
}

// Kasta tärningarna
for (int i = 0; i < dices.length; i++) {
   dices[i].roll();
   System.out.println(dices[i].getDots());
}
```

```
ArrayList<Dice> dices = new ArrayList<Dice>();

// Skapa tärningarna
for (int i = 0; i < 5; i++) {
  dices.add(new Dice());
}

// Kasta tärningarna
for (int i = 0; i < dices.size(); i++) {
  dices.get(i).roll();
  System.out.println(dices.get(i).getDots());
}
```

Anropa följande metoder på listan:

```
/** Lägger in obj sist. */
void add(E obj);

/** Returnerar elementet på plats pos. */
E get(int pos);
```

Inga primitiva typer!

Typargumentet får ***inte*** vara en primitiv typ (`char`, `int`, `boolean`, `double`, …):

```
ArrayList<int> list = new ArrayList<int>();  // Kompileringsfel!
```

Mer om detta vid nästa föreläsning

```java
public class Polygon {
  private ArrayList<Point> vertices;

  /** Skapar en polygon med 0 hörnpunkter. */
  public Polygon() {
    vertices = new ArrayList<Point>();
  }





}
```

Klassen Polygon (labb 8) med en lista istället för en vektor.

```java
public class Polygon {
  private ArrayList<Point> vertices;

  /** Skapar en polygon med 0 hörnpunkter. */
  public Polygon() {
    vertices = new ArrayList<Point>();
  }

  /** Lägger till en ny punkt med koordinaterna x, y. */
  public void addVertex(int x, int y) {
    vertices.add(x, y);
  }

  /** Flyttar polygonen avståndet dx i x-led, dy i y-led. */
  public void move(int dx, int dy) {
    for (int i = 0; i < vertices.size(); i++) {
      vertices.get(i).move(dx, dy);
    }
  }

}
```

Klassen Polygon (labb 8) med en lista istället för en vektor.

Övning: koden innehåller ett vanligt fel. Kan du hitta det?

```java
public class Polygon {
  private ArrayList<Point> vertices;

  /** Skapar en polygon med 0 hörnpunkter. */
  public Polygon() {
    vertices = new ArrayList<Point>();
  }

  /** Lägger till en ny punkt med koordinaterna x, y. */
  public void addVertex(int x, int y) {
    vertices.add(new Point(x, y));
  }
```

Man kan **enbart** lägga in Point-objekt i listan!
Därför måste ett Point-objekt skapas först, som sedan läggs in.

```java
  /** Flyttar polygonen avståndet dx i x-led, dy i y-led. */
  public void move(int dx, int dy) {
    for (int i = 0; i < vertices.size(); i++) {
      vertices.get(i).move(dx, dy);
    }
  }

}
```

Klassen Polygon (labb 8) med en lista istället för en vektor.

Övning: koden innehåller ett vanligt fel. Kan du hitta det?

```java
public class Polygon {
  private ArrayList<Point> vertices;

  /** Skapar en polygon med 0 hörnpunkter. */
  public Polygon() {
    vertices = new ArrayList<Point>();
  }

  ...

  /** Ritar polygonen i fönstret w. */
  public void draw(SimpleWindow w) {
    Point lastPoint = vertices.get(vertices.size()-1);
    w.moveTo(lastPoint.getX(), lastPoint.getY());
    for (int i = 0; i < vertices.size(); i++) {
      Point p = vertices.get(i);
      w.lineTo(p.getX(), p.getY());
    }
  }

}
```

```java
public class Polygon {
  private ArrayList<Point> vertices;

  /** Skapar en polygon med 0 hörnpunkter. */
  public Polygon() {
    vertices = new ArrayList<Point>();
  }

  ...

  /** Tar bort punkten på position pos. Punkterna
   *  numreras från 0 och uppåt i den ordning de
   *  lagts till. */
  public void removeVertex(int pos) {
    vertices.remove(pos);
  }




}
```

Nu är det enkelt att ta bort punkter givet ett index.

Metoden remove anropas på listan.

Övning: räkna antalet ord

```
ArrayList<String> words = new ArrayList<String>();
words.add("apelsin");
words.add("banan");
words.add("druva");
words.add("enbär");
```

ÖVNING: Givet en lista av strängar (ord). Skriv kod för att räkna antalet strängar i listan words som innehåller bokstaven *a* (svar: 3 st).

Allmän iteration av en lista `list`:

```
for (int i = 0; i < list.size(); i++) {
  // Gör något med list.get(i)
}
```

Strängar har metoden:

```
/* Returns true if this string
   contains the String s */
boolean contains(String s);
```

Lösning: räkna antalet ord

```
ArrayList<String> words = new ArrayList<String>();
words.add("apelsin");
words.add("banan");
words.add("druva");
words.add("enbär");

int n = 0;
for (int i = 0; i < words.size(); i++) {
  if (words.get(i).contains("a")) {
    n++;
  }
}
System.out.println(n);
```

ÖVNING: Givet en lista av strängar (ord). Skriv kod för att räkna antalet strängar i listan words som innehåller bokstaven *a* (svar: 3 st).

Allmän iteration av en lista `list`:

```
for (int i = 0; i < list.size(); i++) {
  // Gör något med list.get(i)
}
```

Strängar har metoden:

```
/* Returns true if this string
   contains the String s */
boolean contains(String s);
```

for each-sats

Traversera lista med index-variabel osv:

```
for (int i = 0; i < words.size(); i++) {
  if (words.get(i).contains("a")) {
    n++;
  }
}
```

Vilket kan skrivas om till att använda en *for-each*-sats:

```
// För varje sträng word i listan words
for (String word: words) {
  if (word.contains("a")) {
    n++;
  }
}
```

```java
public class Polygon {
  private ArrayList<Point> vertices;
  ...

  public void move(int dx, int dy) {
    for (int i = 0; i < vertices.size(); i++) {
      vertices.get(i).move(dx, dy);
    }
  }
}
```

```java
public class Polygon {
  private ArrayList<Point> vertices;
  ...

  public void move(int dx, int dy) {
    // ÖVNING:
    // Skriv om metoden move och använd
    // en for-each-sats istället.
  }
}
```

Allmän form av for-each-sats:

```java
for (ElementKlass element: list) {
  // Gör något med element
}
```

Exempel på for-each-sats:

```java
for (String word: words) {
  if (word.contains("a")) {
    n++;
  }
}
```

```java
public class Polygon {
  private ArrayList<Point> vertices;
  ...

  public void move(int dx, int dy) {
    for (int i = 0; i < vertices.size(); i++) {
      vertices.get(i).move(dx, dy);
    }
  }
}
```

```java
public class Polygon {
  private ArrayList<Point> vertices;
  ...

  public void move(int dx, int dy) {
    for (Point p: vertices) {
      p.move(dx, dy);
    }
  }
}
```

Allmän form av for-each-sats:

```java
for (ElementKlass element: list) {
  // Gör något med element
}
```

Exempel på for-each-sats:

```java
for (String word: words) {
  if (word.contains("a")) {
    n++;
  }
}
```

Sammanfattning

- Listor
  - `ArrayList<E>`, generisk. Byt E mot en klass (`Point`, `String`, …)
  - Inte primitive typer! `ArrayList<int>`
  - Metoder: `add`, `get`, `remove`, `size`
  - Inte storleksbegränsad, `size()` ger antal element i listan
- “for each”-satser
  - Ingen index-variabel
  - Temporär variabel istället, med värdet från listan