IN1010 - OBJEKTORIENTERT PROGRAMMERING - GRUPPE 2

VELKOMMEN TIL ANDRE SEMINARTIME

Timen starter 14:15

DAGSORDEN

- Repetisjon
- Oppgaver



REPETISJON

- Kompilering og kjøring
- Variabler, referanser og pekere
- Evaluering av uttrykk
- Parameteroverføring

EKSEMPLER OG OPPGAVER

Hilsen.java

```java
public class Hilsen
{
    public static void main(String[] args)
    {
        System.out.println("Vær hilset!");
    }
}
```

- Hvordan kompilerer man fila?
- Hvordan kjører man den?

```
// javac Hilsen.java for å kompilere
// java Hilsen for å kjøre
```

REPETISJON

- Kompilering og kjøring
- Variabler, referanser og pekere
- Evaluering av uttrykk
- Parameteroverføring

EKSEMPLER OG OPPGAVER

Badeball.java

```java
public class Badeball
{
    private String farge;

    public Badeball(String farge)
    {
        this.farge = farge;
    }

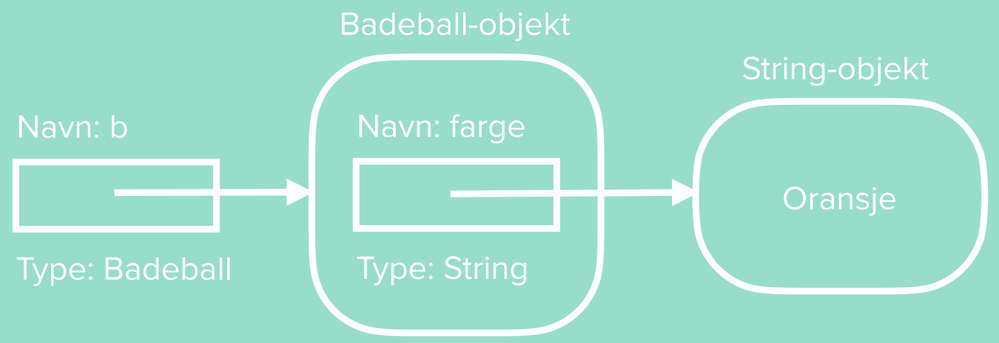
    public static void main(String[] args)
    {
        Badeball b = new Badeball("Oransje");
    }
}
```

- Hvordan tegner man datastrukturen til b?

REPETISJON

- Kompilering og kjøring
- Variabler, referanser og pekere
- Evaluering av uttrykk
- Parameteroverføring

EKSEMPLER OG OPPGAVER





- Hvordan tegner man datastrukturen til b?

REPETISJON

- Kompilering og kjøring
- Variabler, referanser og pekere
- Evaluering av uttrykk
- Parameteroverføring

EKSEMPLER OG OPPGAVER

- Uttrykk kan evalueres til enten:
  1. Primitive typer
  2. Referanseverdier
- Hvordan evalueres uttrykkene under?

```
double tolv = 2 + 10.0;
double tjue = 2 * 10.0;
```

```
// 12.0
// 20.0
```

```
int fem = "2" + 3;
```

```
// feilmelding
```

REPETISJON

- Kompilering og kjøring
- Variabler, referanser og pekere
- Evaluering av uttrykk
- Parameteroverføring

EKSEMPLER OG OPPGAVER

- To typer parametre:
  1. Formelle parametre (parametre)
  2. Aktuelle parametre (argumenter)

- Parameteroverføring skjer slik:
  1. I det en metode kalles evalueres den aktuelle parameteren
  2. Det opprettes en instans av metoden hvor den formelle parameteren er en lokal variabel i metodeinstansen
  3. Det som den aktuelle parameteren evalueres til blir tilordnet variabelen

REPETISJON

- Kompilering og kjøring
- Variabler, referanser og pekere
- Evaluering av uttrykk
- **Parameteroverføring**

EKSEMPLER OG OPPGAVER

Kvadrat.java

```java
public class Kvadrat
{
    private int sidekant;
    private int areal;

    public Kvadrat(int sidekant)
    {
        this.sidekant = sidekant;
        areal = sidekant * sidekant;
    }

    public void endreSidekant(int nySidekant)
    {
        sidekant = nySidekant;
    }
}
```

REPETISJON

- Grensesnitt og implementasjon
- Metodesignaturer
- Observatorer og modifikatorer
- Testing

EKSEMPLER OG OPPGAVER



Kvadrat.java

```java
public class Kvadrat
{
    private int sidekant;
    private int areal;

    public Kvadrat(int sidekant)
    {
        this.sidekant = sidekant;
        areal = sidekant * sidekant;
    }

    public void endreSidekant(int nySidekant)
    {
        sidekant = nySidekant;
    }
}
```

REPETISJON

- Grensesnitt og implementasjon
- Metodesignaturer
- Observatorer og modifikatorer
- Testing

EKSEMPLER OG OPPGAVER

Kvadrat.java

```java
public class Kvadrat
{


    public Kvadrat(int sidekant)
    {


    }

    public void endreSidekant(int nySidekant)
    {

    }
}
```

REPETISJON

- Grensesnitt og implementasjon
- Metodesignaturer
- Observatorer og modifikatorer
- Testing

EKSEMPLER OG OPPGAVER

- Metodesignaturer i Java består av:
  1. Metodenavnet
  2. Typen parametre og deres rekkefølge
- Hører sammen med overloading (senere)
- Hva er metodesignaturen i metoden under?

```
12      public void endreSidekant(int nySidekant)
13      {
14          sidekant = nySidekant;
15      }
```

REPETISJON

- Grensesnitt og implementasjon
- Metodesignaturer
- Observatorer og modifikatorer
- Testing

EKSEMPLER OG OPPGAVER

- Hvilke er observatorer og modifikatorer?

```
12      public void endreSidekant(int nySidekant)
13      {
14          sidekant = nySidekant;      Modifikator
15      }
16
17      public int hentSidekant()       Observator
18      {
19          return sidekant;
20      }
21
22      public int hentAreal()          Observator
23      {
24          return areal;
25      }
```

REPETISJON

- Grensesnitt og implementasjon
- Metodesignaturer
- Observatorer og modifikatorer
- Testing

EKSEMPLER OG OPPGAVER

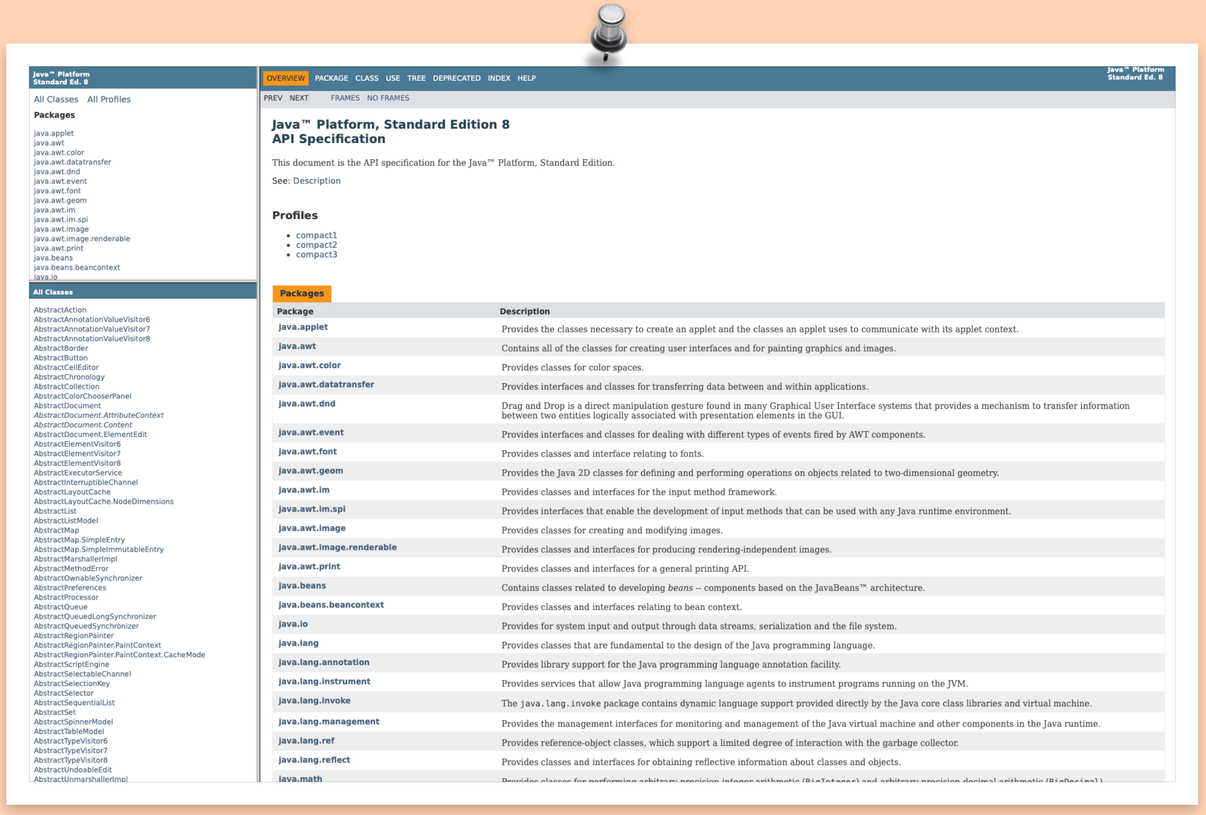
- Enhetstesting og integrasjonstesting
- Hva går galt her?

```
27      public static void main(String[] args)
28      {
29          Kvadrat k = new Kvadrat(2);
30
31          System.out.println(k.hentSidekant());
32          System.out.println(k.hentAreal());
33
34          k.endreSidekant(3);
35
36          System.out.println(k.hentSidekant());
37          System.out.println(k.hentAreal());
38      }
```

REPETISJON

- Java Platform SE 8
- Wrapper-klasser
- String-klassen
- Scanner-klassen
- File-klassen
- PrintWriter-klassen

EKSEMPLER OG OPPGAVER



Java™ Platform Standard Ed. 8

All Classes All Profiles

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
java.awt.im.spi
java.awt.image
java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractAnnotationValueVisitor8
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractChronology
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractElementVisitor8
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimensions
AbstractList
AbstractListModel
AbstractMap
AbstractMap.SimpleEntry
AbstractMap.SimpleImmutableEntry
AbstractMarshallerImpl
AbstractMethodError
AbstractOwnableSynchronizer
AbstractPreferences
AbstractProcessor
AbstractQueue
AbstractQueuedLongSynchronizer
AbstractQueuedSynchronizer
AbstractRegionPainter
AbstractRegionPainter.PaintContext
AbstractRegionPainter.PaintContext.CacheMode
AbstractScriptEngine
AbstractSelectableChannel
AbstractSelectionKey
AbstractSelector
AbstractSequentialList
AbstractSet
AbstractSpinnerModel
AbstractTableModel
AbstractTypeVisitor6
AbstractTypeVisitor7
AbstractTypeVisitor8
AbstractUndoableEdit
AbstractUnmarshallerImpl

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP — Java™ Platform Standard Ed. 8

PREV NEXT FRAMES NO FRAMES

Java™ Platform, Standard Edition 8
API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Profiles

- compact1
- compact2
- compact3

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.
java.awt.geom	Provides the Java 2D classes for defining and performing operations on objects related to two-dimensional geometry.
java.awt.im	Provides classes and interfaces for the input method framework.
java.awt.im.spi	Provides interfaces that enable the development of input methods that can be used with any Java runtime environment.
java.awt.image	Provides classes for creating and modifying images.
java.awt.image.renderable	Provides classes and interfaces for producing rendering-independent images.
java.awt.print	Provides classes and interfaces for a general printing API.
java.beans	Contains classes related to developing *beans* -- components based on the JavaBeans™ architecture.
java.beans.beancontext	Provides classes and interfaces relating to bean context.
java.io	Provides for system input and output through data streams, serialization and the file system.
java.lang	Provides classes that are fundamental to the design of the Java programming language.
java.lang.annotation	Provides library support for the Java programming language annotation facility.
java.lang.instrument	Provides services that allow Java programming language agents to instrument programs running on the JVM.
java.lang.invoke	The `java.lang.invoke` package contains dynamic language support provided directly by the Java core class libraries and virtual machine.
java.lang.management	Provides the management interfaces for monitoring and management of the Java virtual machine and other components in the Java runtime.
java.lang.ref	Provides reference-object classes, which support a limited degree of interaction with the garbage collector.
java.lang.reflect	Provides classes and interfaces for obtaining reflective information about classes and objects.
java.math	Provides classes for performing arbitrary-precision integer arithmetic (`BigInteger`) and arbitrary-precision decimal arithmetic (`BigDecimal`).

REPETISJON

- Java Platform SE 8
- Wrapper-klasser
- String-klassen
- Scanner-klassen
- File-klassen
- PrintWriter-klassen

EKSEMPLER OG OPPGAVER

- Det finnes wrapper-klasser for hver av de primitive typene:
  1. Byte byte
  2. Short short
  3. Integer int
  4. Long long
  5. Float float
  6. Double double
  7. Boolean boolean
  8. Char char
- Wrapper-klassene tilbyr verdifulle metoder

REPETISJON

- Java Platform SE 8
- Wrapper-klasser
- String-klassen
- Scanner-klassen
- File-klassen
- PrintWriter-klassen

EKSEMPLER OG OPPGAVER

1. Gå til String-klassen i Java Platform SE 8
2. Bla ned til metodene
3. Les på beskrivelsene
4. Finn én metode du tror kan være nyttig

REPETISJON

- Java Platform SE 8
- Wrapper-klasser
- String-klasser
- Scanner-klassen
- File-klassen
- PrintWriter-klassen

EKSEMPLER OG OPPGAVER

`nextLine();` `nextInt();` `next();` `nextDouble();`

```
String informasjon = "Henrik 32 1,85 Liker turer i fjellet\n";
Scanner les = new Scanner(informasjon);

String navn = les.

int alder = les.

double hoyde = les.

les.skip(" ");

String bio = les.

les.close();

System.out.print("Profil\n");
System.out.print(navn + ", " + alder + " år gammel, ");
System.out.println(hoyde + " meter høy");
System.out.println(bio);
```

REPETISJON

- Java Platform SE 8
- Wrapper-klasser
- String-klassen
- **Scanner-klassen**
- File-klassen
- PrintWriter-klassen

EKSEMPLER OG OPPGAVER

`nextLine();` `nextInt();` `next();` `nextDouble();`

```
String informasjon = "Henrik 32 1,85 Liker turer i fjellet\n";
Scanner les = new Scanner(informasjon);

String navn = les.next();

int alder = les.nextInt();

double hoyde = les.nextDouble();

les.skip(" ");

String bio = les.nextLine();

les.close();

System.out.print("Profil\n");
System.out.print(navn + ", " + alder + " år gammel, ");
System.out.println(hoyde + " meter høy");
System.out.println(bio);
```

REPETISJON

- Java Platform SE 8
- Wrapper-klasser
- String-klassen
- Scanner-klassen
- File-klassen
- PrintWriter-klassen

EKSEMPLER OG OPPGAVER

- Først kan en opprette et File-objekt
- Så kan en sende fila til et Scanner-objekt
- Her er det viktig å sjekke om det finnes noe å skanne før man gjør det!



```
File fil = new File("tekstfil.txt");
Scanner les = new Scanner(fil);

les.hasNext();
les.hasNextLine();

les.hasNextInt();
les.hasNextDouble();
```

REPETISJON

- Java Platform SE 8
- Wrapper-klasser
- String-klassen
- Scanner-klassen
- File-klassen
- PrintWriter-klassen

EKSEMPLER OG OPPGAVER

- For å skrive ut til fil kan en benytte et PrintWriter-objekt
- Man sender med filnavnet det skal skrives til, og så kan en bruke print-, println- og printf-metodene for å skrive til filen
- Husk å lukke PrintWriter-objektet med close-metoden



```
PrintWriter skriv = new PrintWriter("profil.txt");

skriv.print("Profil\n");
skriv.print(navn + ", " + alder + " år gammel, ");
skriv.println(hoyde + " meter høy");
skriv.println(bio);

skriv.close();
```

REPETISJON

- Array
- ArrayList
- HashMap
- For-løkker
- Forenklede for-løkker

EKSEMPLER OG OPPGAVER

Syntax

To construct an array: `new typeName[length]`

To access an element: `arrayReference[index]`

Name of array variable

Element type

Length

Type of array variable

```
double[] values = new double[10];

double[] moreValues = { 32, 54, 67.5, 29, 35 };
```

List of initial values

Use brackets to access an element.

```
values[i] = 0;
```

The index must be ≥ 0 and < the length of the array.

See page 255.

REPETISJON

- Array
- ArrayList
- HashMap
- For-løkker
- Forenklede for-løkker

EKSEMPLER OG OPPGAVER

Table 1 Declaring Arrays	
`int[] numbers = new int[10];`	An array of ten integers. All elements are initialized with zero.
`final int LENGTH = 10;` `int[] numbers = new int[LENGTH];`	It is a good idea to use a named constant instead of a "magic number".
`int length = in.nextInt();` `double[] data = new double[length];`	The length need not be a constant.
`int[] squares = { 0, 1, 4, 9, 16 };`	An array of five integers, with initial values.
`String[] friends = { "Emily", "Bob", "Cindy" };`	An array of three strings.

```
int[] tom = new int[10];     // 0 0 0 0 0 0 0 0 0 0
tom[5] = 5;                  // 0 0 0 0 0 5 0 0 0 0

// starter en array med 0 1 2 3 4 5 6 7 8 9
int[] titalls = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

REPETISJON

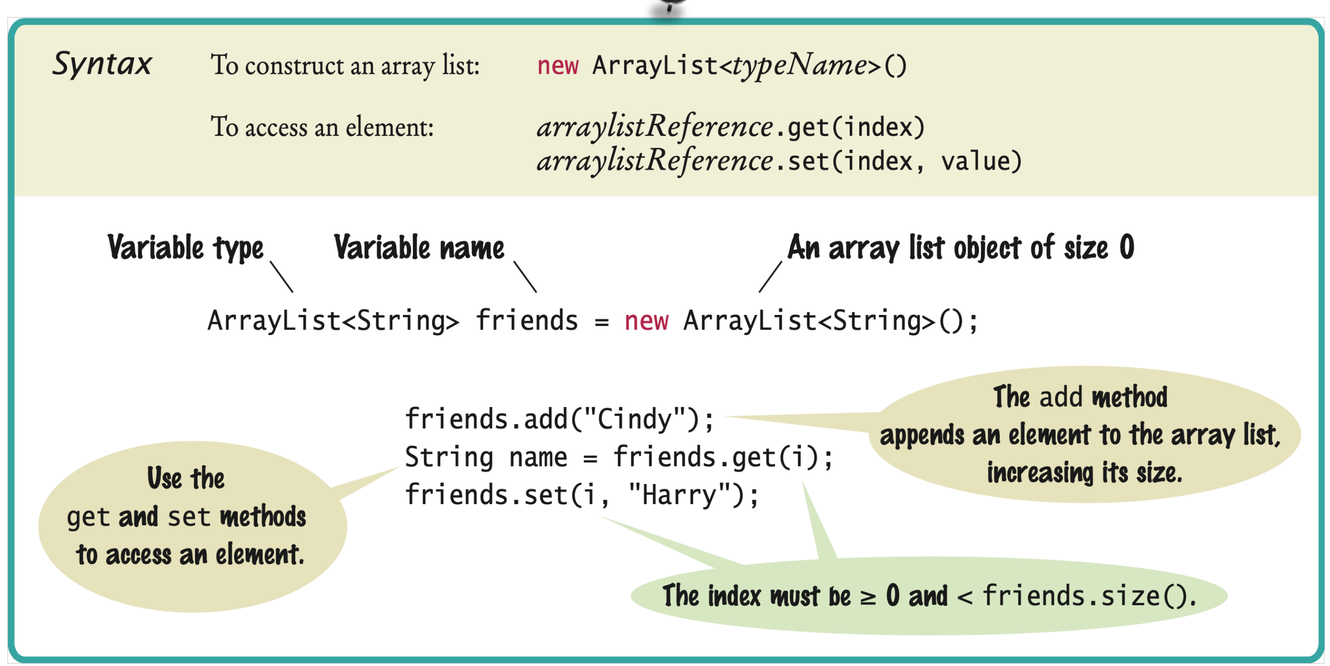
- Array
- ArrayList
- HashMap
- For-løkker
- Forenklede for-løkker

EKSEMPLER OG OPPGAVER

Syntax

To construct an array list: `new ArrayList<typeName>()`

To access an element: *arraylistReference*`.get(index)`
arraylistReference`.set(index, value)`



REPETISJON

- Array
- ArrayList
- HashMap
- For-løkker
- Forenklede for-løkker

EKSEMPLER OG OPPGAVER



Table 2 Working with Array Lists

`ArrayList<String> names = new ArrayList<String>();`	Constructs an empty array list that can hold strings.
`names.add("Ann");` `names.add("Cindy");`	Adds elements to the end.
`System.out.println(names);`	Prints `[Ann, Cindy]`.
`names.add(1, "Bob");`	Inserts an element at index 1. `names` is now `[Ann, Bob, Cindy]`.
`names.remove(0);`	Removes the element at index 0. `names` is now `[Bob, Cindy]`.
`names.set(0, "Bill");`	Replaces an element with a different value. `names` is now `[Bill, Cindy]`.
`String name = names.get(i);`	Gets an element.
`String last = names.get(names.size() - 1);`	Gets the last element.

REPETISJON

- Array
- ArrayList
- HashMap
- For-løkker
- Forenklede for-løkker

EKSEMPLER OG OPPGAVER

- En HashMap kan tenkes på som en ordbok, altså at en har nøkkel- og innholdsverdier som er knyttet sammen i par



```java
HashMap<String, String> karakterer = new HashMap<String, String>();

karakterer.put("IN1000", "A");
karakterer.put("IN1010", "C");

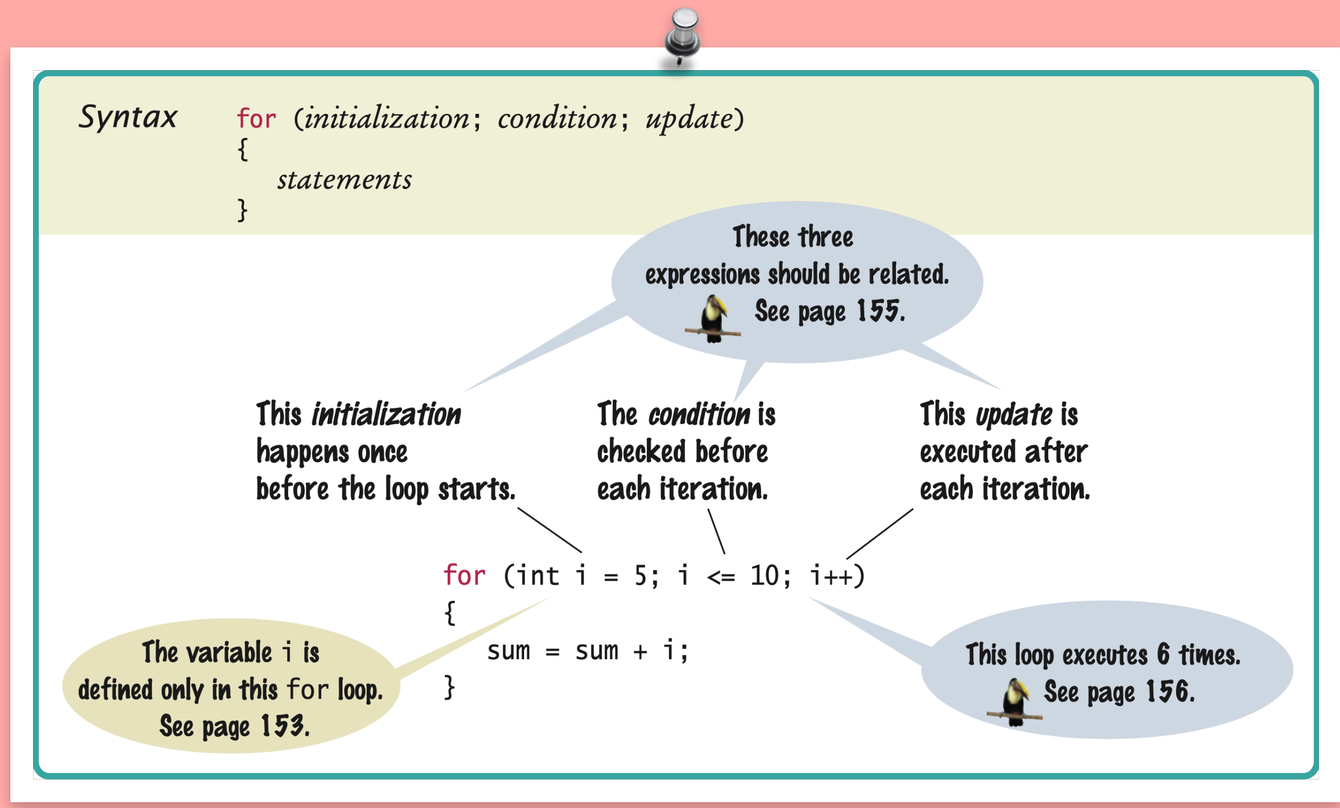
// diamant-syntaks
HashMap<String, String> adresser = new HashMap<>();
```

- Bruk Java Platform SE 8 til å finne:
  1. Hvordan importere HashMap
  2. Én eller flere nyttige metoder

REPETISJON

- Array
- ArrayList
- HashMap
- For-løkker
- Forenklede for-løkker

EKSEMPLER OG OPPGAVER



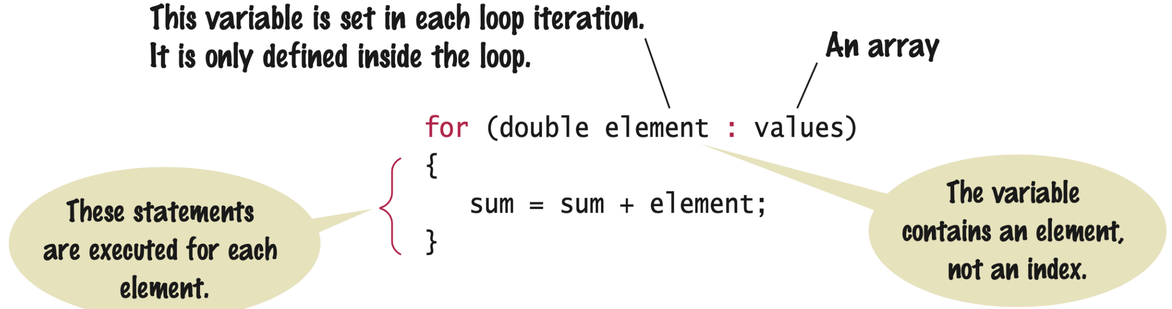
REPETISJON

- Array
- ArrayList
- HashMap
- For-løkker
- Forenklede for-løkker

EKSEMPLER OG OPPGAVER

Syntax

```
for (typeName variable : collection)
{
   statements
}
```



VI SEES

HADE 👋