IN1010 - OBJEKTORIENTERT PROGRAMMERING

VELKOMMEN TIL ANDRE REPETISJONSTIME

Timen starter 12:15

REPETISJON

- Primitive typer
- Java Platform SE 8
- Evaluering av uttrykk
- Tilgangsmodifikatorer

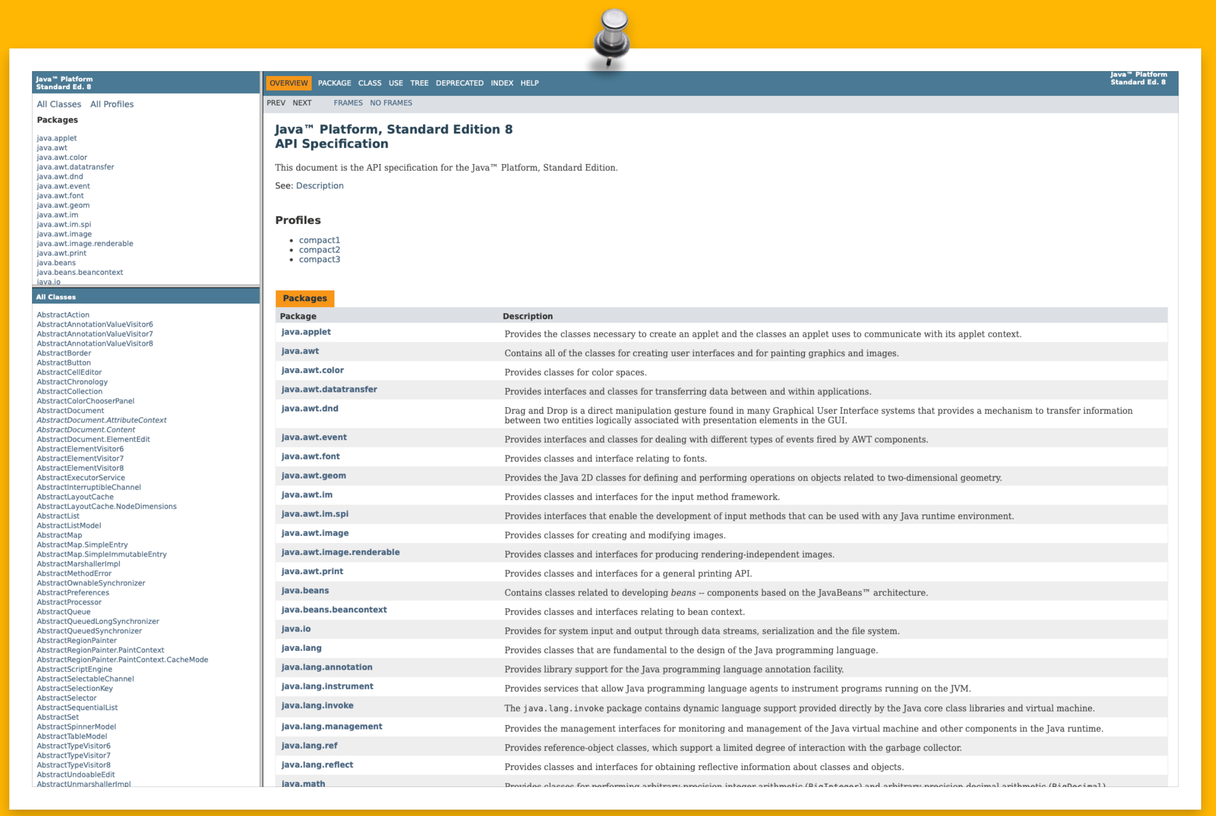
EKSEMPLER

- Det finnes åtte primitive typer med tilhørende wrapper-klasser:
  - byte Byte
  - short Short
  - int Integer
  - long Long
  - float Float
  - double Double
  - boolean Boolean
  - char Character
- Wrapper-klassene tilbyr verdifulle metoder

REPETISJON

- Primitive typer
- Java Platform SE 8
- Evaluering av uttrykk
- Tilgangsmodifikatorer

EKSEMPLER



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

- Primitive typer
- Java Platform SE 8
- Evaluering av uttrykk
- Tilgangsmodifikatorer

EKSEMPLER

- Uttrykk kan evalueres til enten:
  - Primitive typer
  - Referanseverdier
- Alt som peker på noe annet enn en primitiv type er en referanseverdi

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

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

REPETISJON

- Primitive typer
- Java Platform SE 8
- Evaluering av uttrykk
- Tilgangsmodifikatorer

EKSEMPLER

- Uttrykk kan evalueres til enten:
  - Primitive typer
  - Referanseverdier
- Alt som peker på noe annet enn en primitiv type er en referanseverdi

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

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

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

REPETISJON

- Primitive typer
- Java Platform SE 8
- Evaluering av uttrykk
- Tilgangsmodifikatorer

EKSEMPLER

- Uttrykk kan evalueres til enten:
  - Primitive typer
  - Referanseverdier
- Alt som peker på noe annet enn en primitiv type er en referanseverdi

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

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

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

```
// feilmelding
```

REPETISJON

- Primitive typer
- Java Platform SE 8
- Evaluering av uttrykk
- Tilgangsmodifikatorer

EKSEMPLER

- Det finne fire tilgangsmodifikatorer:
  - public
  - protected
  - <ingenting> ("package private")
  - private

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	X
no modifier	Y	Y	X	X
private	Y	X	X	X

REPETISJON

- Primitive typer
- Java Platform SE 8
- Evaluering av uttrykk
- **Tilgangsmodifikatorer**

EKSEMPLER

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

- Primitive typer
- Java Platform SE 8
- Evaluering av uttrykk
- **Tilgangsmodifikatorer**

EKSEMPLER

Kvadrat.java

```
1  public class Kvadrat
2  {
3
4
5
6      public Kvadrat(int sidekant)
7      {
8
9
10     }
11
12     public void endreSidekant(int nySidekant)
13     {
14
15     }
16 }
```

REPETISJON

- Metodesignaturer
- Observator og modifikator
- Testing
- Unntak og unntakshåndtering

EKSEMPLER

- Metodesignaturer i Java består av:
  - Metodenavnet
  - Typen parametre og deres rekkefølge
- Dette muliggjør overlasting, altså det å ha flere metoder med samme navn i en gitt klasse

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

REPETISJON

- Metodesignaturer
- Observator og modifikator
- Testing
- Unntak og unntakshåndtering

EKSEMPLER



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

REPETISJON

- Metodesignaturer
- Observator og modifikator
- Testing
- Unntak og unntakshåndtering

EKSEMPLER

```
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

- Metodesignaturer
- Observator og modifikator
- Testing
- Unntak og unntakshåndtering

EKSEMPLER

- Enhetstesting er å teste hver enkelt enhet i koden
- For eksempel en metode

- Integrasjonstesting er å teste at enhetene fungerer sammen
- For eksempel at to objekter av forskjellige klasser påvirker hverandre på forventet måte

- Det kommer mer om testing senere i semesteret

REPETISJON

- Metodesignaturer
- Observator og modifikator
- Testing
- Unntak og unntakshåndtering

EKSEMPLER

- Handler om å oppdage feil og håndtere feil
- Feil oppdaget: throw new unntak
- Velg passende type feilmelding
- Programmet stopper opp der hvor unntaket kastes

- Håndtere feil: bruk catch
- Catch spesifikt først, og så mer generelt
- Ingen handler gir feilmelding og programmet avsluttes
- Kan kaste videre med throws

REPETISJON

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

`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

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

`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.

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

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

`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.

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

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

`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.

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

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

`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

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

- For å lese en fil oppretter en et File-objekt
- File-objektet kan sendes til en Scanner
- 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

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

- 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

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

- En ArrayList likner på en liste i Python
- Størrelsen øker når en legger til elementer
- Holder kun på elementer av samme type

Syntax

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

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

Variable type — Variable name — An array list object of size 0

```
ArrayList<String> friends = new ArrayList<String>();
```

```
friends.add("Cindy");
String name = friends.get(i);
friends.set(i, "Harry");
```

Use the `get` and `set` methods to access an element.

The `add` method appends an element to the array list, increasing its size.

The index must be ≥ 0 and < `friends.size()`.

REPETISJON

- Scanner
- File
- PrintWriter
- ArrayList
- HashMap

EKSEMPLER

- En HashMap kan tenkes på som en ordbok
- Nøkkelverdier knyttes mot innholdsverdier
- Må oppgi typen for nøkkelverdi og innholdsverdi

- Legg merke til diamantsyntaks



```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<>();
```

VIL DU HA NOE REPETERT?

SEND ØNSKER TIL LILARSSE@IFI.UIO.NO

VI SEES

HADE 👋