Contents

[Contents 1](#_Toc435100013)

[Dag 1 klasser og objekter. 3](#_Toc435100014)

[public class Bike 3](#_Toc435100015)

[public class RunClient 4](#_Toc435100016)

[Dag 2 arv og interface 4](#_Toc435100017)

[Dag 3 kildekode, datatyper, primitive typer, operatorer 5](#_Toc435100018)

[Operatorer på primitive datatyper 10](#_Toc435100019)

[Pass-by-value og pass-by-reference 11](#_Toc435100020)

[Dag 4: arrays og loops 11](#_Toc435100021)

[public class RunClient 11](#_Toc435100022)

[public class BikeShop 12](#_Toc435100023)

[public class Bike 12](#_Toc435100024)

[public class ArrayTests 13](#_Toc435100025)

[Dag 5 array, loop, switch 14](#_Toc435100026)

[public class Switch 14](#_Toc435100027)

[Dag 6 Arrays, loop, user input 14](#_Toc435100028)

[public class UserInput 14](#_Toc435100029)

[public class RunClientUser 15](#_Toc435100030)

[public class RunClient 15](#_Toc435100031)

[public class Phone 16](#_Toc435100032)

[public class IPhone 16](#_Toc435100033)

[public class Controler 16](#_Toc435100034)

[public class CommandLine 17](#_Toc435100035)

[public class Android 17](#_Toc435100036)

[Dag 7 Strings and ArrayList 18](#_Toc435100037)

[Et fedt citat om læring 18](#_Toc435100038)

[Object oriented stuff 19](#_Toc435100039)

[String 1 20](#_Toc435100040)

[String 2 20](#_Toc435100041)

[ArrayList 21](#_Toc435100042)

[Dag 8 21](#_Toc435100043)

[public class TestFinal 21](#_Toc435100044)

[Dag 9. Final 23](#_Toc435100045)

[public class UtilChild 23](#_Toc435100046)

[public /\*final\*/ class Util 24](#_Toc435100047)

[public class RunClient 24](#_Toc435100048)

[public class Paint 25](#_Toc435100049)

[public class Finalist 25](#_Toc435100050)

[Dag 10. Ensum 25](#_Toc435100051)

[public class Test 25](#_Toc435100052)

[public class Run 26](#_Toc435100053)

[public enum WeekDays 26](#_Toc435100054)

[Dag 13. Innerclass 26](#_Toc435100055)

[public class Container 26](#_Toc435100056)

[public class Run 27](#_Toc435100057)

[Dag 14 I\_O exception 28](#_Toc435100058)

[public class ByteExample 28](#_Toc435100059)

[Dag 15 Streams 2 31](#_Toc435100060)

[public class ByteStream 31](#_Toc435100061)

[public class CharacterStream 32](#_Toc435100062)

[public class BufferedStream 32](#_Toc435100063)

[public class BridgeStream 33](#_Toc435100064)

[public class DataStreams 34](#_Toc435100065)

[Dag 16 Strams 3 35](#_Toc435100066)

[public class CountryListSerialize 35](#_Toc435100067)

[public class Country 37](#_Toc435100068)

[Dag 17 java.nio 37](#_Toc435100069)

[public class PathStuff 37](#_Toc435100070)

[public class VerifyFile 38](#_Toc435100071)

[Dag 18 Repitation java.nio 39](#_Toc435100072)

[public class VerifyFile 39](#_Toc435100073)

[Dag 19 repitation 40](#_Toc435100074)

[Øvelser med ArrayList 40](#_Toc435100075)

[public abstract class Animal 40](#_Toc435100076)

[public class Cat 40](#_Toc435100077)

[public class Head 41](#_Toc435100078)

[public class Lion 41](#_Toc435100079)

[public class RunClient 41](#_Toc435100080)

[public class DayTest 42](#_Toc435100081)

[public enum Days 42](#_Toc435100082)

[public class Day 42](#_Toc435100083)

[Dag 20 Throw 43](#_Toc435100084)

[public class ThrowExample 43](#_Toc435100085)

[public class ReferenceExample 43](#_Toc435100086)

[public class Person 44](#_Toc435100087)

[public class Game 44](#_Toc435100088)

[public interface Perpetual 44](#_Toc435100089)

[public class RunClient 44](#_Toc435100090)

[public class Rust 44](#_Toc435100091)

[public class Tekken3 45](#_Toc435100092)

[public interface Winnable 45](#_Toc435100093)

[Dag 20 wrapper class 45](#_Toc435100094)

[public class WrapperStuff 45](#_Toc435100095)

# Dag 1 klasser og objekter.

<https://www.youtube.com/watch?v=tYRrI9cnZ_E&list=PLQtGc-vjIOdzGq_mYkCq4ShiwEVyruo7Z>

public class Bike{

private String color; //default = null

public int gear; //default = 0

private int weels; //default = 0

//public/private er access modifiers

public Bike(String color){

this.color = color;

}

//constructor

public Bike(int gear){

this.gear = gear;

}

//empty constructor

public Bike(){}

//This is how you write a method:

//access modifier retur type navn parameter

public void ride(){

System.out.println("I'm riding yeah");

System.out.println("I'm still riding");

}

public void setGear(int gear){

this.gear = gear;

}

public int getGear(){

return gear;

}

}

public class RunClient{

//Dette er en én linjes kommentar

/\* Dette er flere linjers kommentar

main metoden er der hvor dit program starter fra

\*/

public static void main(String[] args){

//type variabelnavn = "new" konstruktørens navn(parametre)

//here we create an object

Bike b = new Bike();

/\*this creates a null-pointer exception.

because b is null and we try to call a method on it

you can try to uncomment the comment to see it bee thrown

\*/

//b = null;

//b1.setGear(21);

//her kalder vi ride() metoden på Bike b

b.ride();

b.setGear(9);

System.out.println("The bike has "+b.getGear()+" gears");

}

}

# Dag 2 arv og interface

# Dag 3 kildekode, datatyper, primitive typer, operatorer

Recap

Intro til Javas kildekode ● Hvad er datatyper?

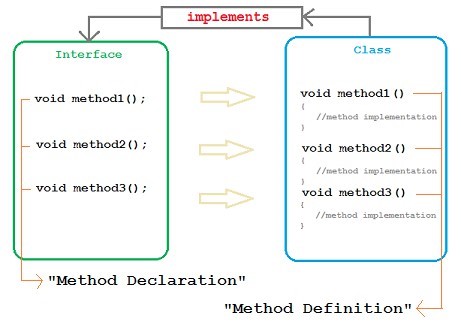
Primitive datatyper i Java

Øvelser med primitive datatyper

Pass-by-value og pass-by-reference

Øvelser med pass-by-value og pass-by-reference

Interface



Intro til Javas kildekode

For Unix, Windows 7 og 8: Åben terminalen

Skriv: java -verbose

Se hvilken path som jre/ har.

Naviger til jre/lib/ igennem terminalen, skriv: jar tf rt.jar

se at /java/lang/Object.class er der.

skriv javap java/lang/Object

En gui kan også bruges, f.eks. JD-GUI.

Hvad er datatyper?

En datatype er et sæt værdier og et sæt operationer man kan foretage på de værdier.

int som eksempel:

Værdier imellem -231 og 231 – 1.

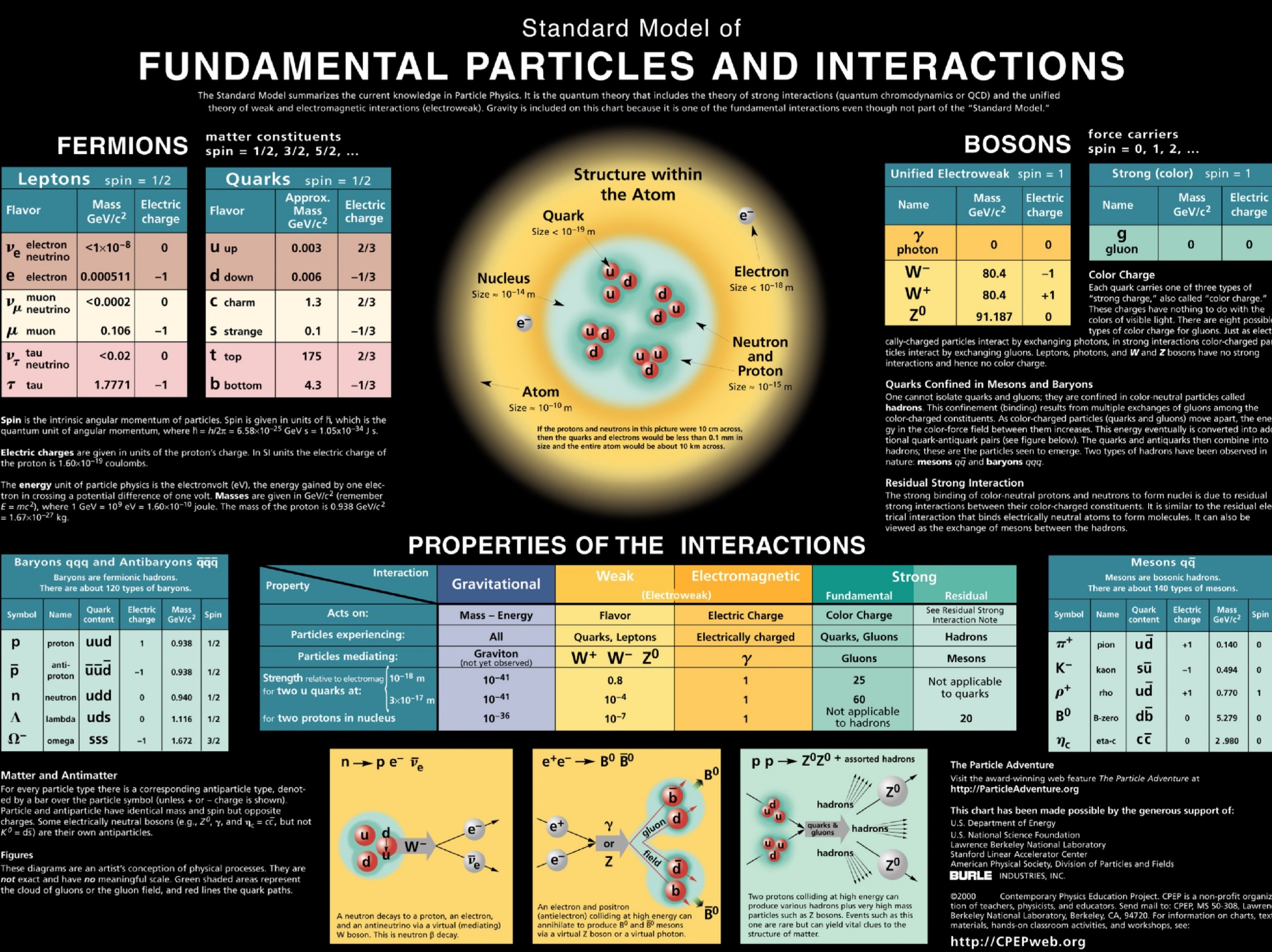
Operationer er basal aritmetik

Bike som eksempel

Værdier af typen String i color

Operationerne ride() og setColor()

Det samlede sæt operationer kalder man API (Application Programming Interface)



2 datatyper i Java

Primitive typer / primitive types

Reference typer / reference types / klasser

Primitive typer: – 8 typer, kun i Java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| byte 8 bits | {-128;127} | | byte b = 100; | |
| char 16 bits | {0; 216-1} | char f = 'f'; | |
| short 16 bits | {-215-1; 215-1} | short s = 32760 | |
| int 32 bits | {-231; 231-1} | int i = 152342; | |
| long 64 bits  ● Floating-points: | {-263; 263-1} | long l = 3426L | |
| float 32 bits | {-2149;(2\*2-23)\*2-127} | float f = 23.34f | |
| double 64 bits ● Binary: | {-21074;(2-2-52)\*21023} | double d=34.221 | |
| boolean 1 bit | sand / falsk | boolean b = true | |

Reference typer:

Vores egne klasser, og tusindevis af klasser i Javas bibliotek.

Primitive typer i Java

● Integers:

|  |  |
| --- | --- |
| unary postfix | expr++ expr-- |
| Unary | ++expr --expr +expr -expr ~ ! |
| Multiplicative | \* / % |
| Additive | + - |
| Shift | << >> >>> |
| Relational | < > <= >= instanceof |
| Equality | == != |
| bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | | |
| logical AND | && |
| logical OR | || |
| Ternary | ? : |
| assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |

# Operatorer på primitive datatyper

Syntaks for at lave et objekt

Object o; o = new Object();

**Declaration/Erklæring**: Øverste linje erklærer objektet

**Instantiation/instantiering:** 'new' instantierer et nyt objekt.

**Initialization/Initialisering**: Konstruktøren intialiserer objektet, enten ved en default konstruktør eller ved at give det nogle værdier.

# Pass-by-value og pass-by-reference

* Når primitive typer kommer ind i en metode kopieres den værdi, der ligger i adressen.
* Når objekter kommer ind i en metode, kopieres deres addresse i memory. Det vil sige: En variabel har en reference til et sted i hukommelsen, hvor det objekt ligger. Værdien af den reference sendes ind i en metode.

# Dag 4: arrays og loops

public class RunClient{

//Dette er en én linjes kommentar

/\* Dette er flere linjers kommentar

main metoden er der hvor dit program starter fra

\*/

public static void main(String[] args){

//type variabelnavn = "new" konstruktørens navn(parametre)

//here we create an object

Bike b = new Bike();

b.setGear(9);

b.setColor("blue");

BikeShop bikeShop = new BikeShop(10);

System.out.println(bikeShop.getBikes().length);

bikeShop.addBike(b, 5);

bikeShop.printBikes();

}

}

public class BikeShop{

private Bike[] bikes;

public BikeShop(int number){

bikes = new Bike[number];

}

public void addBike(Bike b, int i){

//et index til at placere b

bikes[i] = b;

//add my bike to bikes

}

public void printBikes(){

//looper igennem bikes og printer hver enkelt

for(int i = 0; i < bikes.length; i++){

System.out.println(bikes[i]);

}

}

//[] betyder array

public Bike[] getBikes(){

return bikes;

}

}

public class Bike{

private String color; //default = null

public int gear; //default = 0

private int weels = 2;

//public/private er access modifiers

public Bike(String color){

this.color = color;

}

public void setColor(String color){

this.color = color;

}

//constructor

public Bike(int gear){

this.gear = gear;

}

//empty constructor

public Bike(){}

//This is how you write a method:

//access modifier retur type navn parameter

public void ride(){

System.out.println("I'm riding yeah");

System.out.println("I'm still riding");

}

public void setGear(int gear){

this.gear = gear;

}

public int getGear(){

return gear;

}

public String toString(){

return " color: "+color+", weels: "+weels;

}

}

public class ArrayTests{

public static void main(String[] args){

int[] a = new int[5];

//int a[45];

//man angiver ikke antal elementer

int[] myArray = {324, 23, 56, 457};

//for-loop

for(int i = 0; i < a.length; i++){

a[i] = i+1;

}

for(int i = 0; i < a.length; i++){

System.out.println(a[i]);

}

for(int i = 4; i <= 0; i--){

System.out.println(a[i]);

}

//for-each loop

System.out.println("This is the enhanced for loop / for-each loop");

for(int i: a){

System.out.println(i);

}

//while loop

System.out.println("This is my while loop:");

int c = 0;

while(c < a.length){

System.out.println(a[c]);

c++;

}

//do-while loop

System.out.println("And here is the do - while loop");

int d = 0;

do{

System.out.println(" YEah: " + a[d]);

d++;

} while(d > a.length);

}

}

# Dag 5 array, loop, switch

public class Switch{

//

//switch minder om else og if

public static void main(String[] args){

String name = null;

String person = "MoTHEr";

//person = person.toLowerCase();

switch(person.toLowerCase()){

case "father": {

name = "Lars";

break;

}

case "brother": {

name = "Andreas";

break;

}

case "mother":{

name = "Anette";

System.out.println("something else happened");

}

break;

default:

name = "INVALID PERSON";

}

System.out.println(name);

}

}

# Dag 6 Arrays, loop, user input

import java.util.Scanner;

public class UserInput{

private Scanner scanner;

private String name;

public UserInput(){

scanner = new Scanner(System.in);

}

public void receiveInput(){

System.out.println("What's your name?");

name = scanner.nextLine();

System.out.println("Your input: "+name);

switch(name){

case "Ben":

System.out.println("cool name");

break;

case "Jerry":

System.out.println("So like Springer...?");

break;

default:

System.out.println("I don't know you");

break;

}

}

}

public class RunClientUser{

public static void main(String[] args){

UserInput userInput = new UserInput();

userInput.receiveInput();

}

}

public class RunClient{

public static void main(String[] args){

//laver et array af superklassen

Phone[] phones = new Phone[5];

IPhone iPhone1 = new IPhone();

IPhone iPhone2 = new IPhone();

IPhone iPhone3 = new IPhone();

Android android1 = new Android();

Android android2 = new Android();

phones[0] = iPhone1;

phones[1] = iPhone2;

phones[2] = iPhone3;

phones[3] = android1;

phones[4] = android2;

System.out.println(phones[2]);

Controler cont = new Controler();

IPhone[] iPhones = cont.getIPhones(phones);

System.out.println(iPhones.length);

}

}

public class Phone{

public Phone(){}

}

//arver fra: Phone

public class IPhone extends Phone{

public IPhone(){

//super.doSomething();

//super.myColor;

super();

}

}

public class Controler{

//ting vi skal bruge:

//instanceof

//loop

public IPhone[] getIPhones(Phone[] phones){

//finde ud af hvor mange Iphones der er i arrayet

int count = 0;

//

for(int i = 0; i < phones.length; i++){

if(phones[i] instanceof IPhone){

count++;

}

}

//oprette et ny array med plads til dem

IPhone[] iPhones = new IPhone[count];

//flytte hvert enkelt IPhone objekt fra phones

//til vores nye array

int index = 0;

for(int i = 0; i < phones.length; i++){

if(phones[i] instanceof IPhone){

//

iPhones[index] = (IPhone)phones[i];

index++;

}

}

return iPhones;

//returnerer det nye array

}

}

public class CommandLine{

public static void main(String[] argentina){

int result = Integer.parseInt(argentina[1])+10;

System.out.println("my args: "+argentina[0] + " "+result);

methodOne();

//methodOne(false);

methodTwo(12, 235, 346, 457, 6);

}

//overloading

public static void methodOne(boolean b){

System.out.println("mehtod 1");

}

public static void methodOne(){

System.out.println("mehtod 2");

}

//varargs

public static void methodTwo(int...ints){

for(int i: ints){

System.out.println(i);

}

}

}

public class Android extends Phone{

//samme som default konstruktør

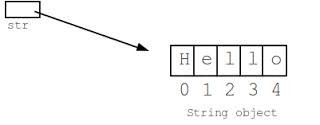
public Android(){

super();

}

}

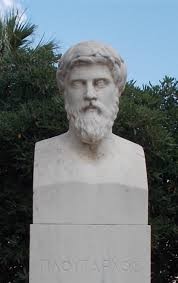
# Dag 7 Strings and ArrayList



* Recap
* String
* ArrayList

# Et fedt citat om læring

”the correct analogy for the mind *is not a vessel that needs filling*, but wood that needs igniting — no more — and then it motivates one towards originality and instills the desire for truth”

Plutarch (år 50 - 120)

# Object oriented stuff

objekter for-each loop import metoder arv værdi interfaces Java bibliotek klasser wrapper-typer variabler break assignment String boolean instans variabler int Integer primitive typer relationelle operatorer .java filer main-metode parameter klasse variabler String literal arrays loops double superklasse while-loop konstruktører datatyper subklasse initialisere unary operatorer .class filer kompiler instantiere

# String 1

String = ”Hello World”; //” ” er en string literal

Når compileren møder en string literal, så laver den et nyt String objekt. Så

String str = "abc";

er det samme som

char data[] = {'a', 'b', 'c'};

String str = new String(data);

# String 2

* Strings er immutable

Betyder: Værdien for et String objekt kan ikke ændres når det først er oprettet.

* Hvad sker der så her? String s = ”abc”; s = s.toUpperCase(); //s er nu ”ABC”?

# ArrayList

* Et array med en masse ekstra lir ● Kan ikke indeholde primitive typer. ● Men wrapper typer er o.k.

15 minutter:

* Hvilke interfaces implementerer ArrayList?
* Hvilke klasser arver ArrayList fra?
* Hvad gør add(), get(), size(), remove(), sort()?
* Hvad returnerer indexOf() hvis den ikke kan finde det man giver den som parameter?

# Dag 8

public class TestFinal{

public static void main(String[]args){

//final variable

//What does it say about variables that they are declared final?

//Why would we want to have some variables being declared final?

final Test1 t = new Test1(23);

System.out.println(t.getNumber());

System.out.println(t.addSixteenResult());

//why does this give a compile time error?

//t = new Test1(12);

}

}

class Test{

private int number;

public Test(int number){

this.number = number;

}

public int getNumber(){

return number;

}

public void setNumber(int number){

this.number = number;

}

//final method

//What does it say about a method that it is declared final?

//What are some use-cases?

final public void addFive(){

number += 5;

}

}

//final class

//What does it say about a class that it is declared final?

//What are some use-cases?

final class Test1 extends Test{

public Test1(int number){

super(number);

}

//Is this an OO use of a final method?

public int addSixteenResult(){

addFive();

addFive();

addFive();

setNumber(getNumber()+1);

return getNumber();

}

}

Øvelser med Strings

.length()

String dokumentation.

Sværhedsgrad 1

//Hello og parameteret og ! returneres

helloName("Bob") → "Hello Bob!"

+” !”+

//halvdelen af strengen returneres

firstHalf("WooHoo") → "Woo"

//de første to chars i strengen returneres

twoChar("java") → "ja"

//String objektet vendes om

rotate(”great!”) → ”!taerg”

Sværhedsgrad 2

//Return true hvis en given char indeholder ”bob”,

men hvor det middterste 'o' char kan være enhver

char.

bobThere("abcbob") → true

bobThere("b9b") → true

bobThere("bac") → false

//Returner en streng hvor parameteret, hvis det

indeholder en stjerne, så slet hver bogstav der står

# Dag 9. Final

public class UtilChild extends Util{

public String toString(){

return "I'm the child of a final class";

}

}

import java.util.ArrayList;

public /\*final\*/ class Util{

//tage en arrayliste og en farve ind TJEK

//lede arraylisten igennem og lægge alle objekter med farven over i en ny arrayliste TJEK

//returnere den nye arrayliste

public ArrayList<Paint> getPaintsByColor(ArrayList<Paint> paints, String color){

ArrayList<Paint> newPaints = new ArrayList<Paint>();

for(Paint p: paints){

if(p.getColor().equalsIgnoreCase(color)){

newPaints.add(p);

}

}

return newPaints;

}

//final

public /\*final\*/ String toString(){

return "I'm the util class";

}

}

import java.util.ArrayList;

public class RunClient{

public static void main(String[] args){

Paint p1 = new Paint("Blue");

Paint p2 = new Paint("Red");

Paint p3 = new Paint("Yellow");

Paint p4 = new Paint("Yellow");

ArrayList<Paint> paints = new ArrayList<Paint>();

paints.add(p1);

paints.add(p2);

paints.add(p3);

paints.add(p4);

Util u = new Util();

ArrayList<Paint> result = u.getPaintsByColor(paints, "yellow");

for(Paint p: result){

System.out.println(p);

}

}

}

import java.time.\*;

public class Paint{

private String color;

private LocalDate date;

public Paint(String color){

this.color = color;

}

public String getColor(){

return color;

}

public String toString(){

return "my color is : "+color;

}

}

public class Finalist{

public static void main(String[] args){

UtilChild uc = new UtilChild();

//final

final int var = 4;

var = 5;

}

}

# Dag 10. Ensum

public class Test{

private static String name = "Ovid";

public static String getName(){

return name;

}

enum WeekDays{

MONDAY,

TUESDAY;

}

//compareTo()

//getDeclaringClass()

//name()

//ordinal()

//toString()

//valueOf()

public static void main(String[] args){

String answer = "Tuesday";

System.out.println(WeekDays.valueOf(answer.toUpperCase()));

System.out.printf("\n");

print(WeekDays.MONDAY);

System.out.println(WeekDays.TUESDAY.ordinal());

System.out.println(WeekDays.TUESDAY.getDeclaringClass());

System.out.println(WeekDays.TUESDAY.name());

System.out.println(WeekDays.TUESDAY.toString());

//System.out.println(WeekDays.valueOf(WeekDays, "TUESDAY"));

}

public static void print(WeekDays info){

//WeekDays.TUESDAY

System.out.println(info.TUESDAY);

switch(info){

case MONDAY:

System.out.println("Great, a new week!");

break;

case TUESDAY:

System.out.println("When is it weekend...??");

break;

}

}

}

public class Run{

public static void main(String[] args){

Test t = new Test();

System.out.println("Author of Metamorphosis: "+Test.getName());

}

}

public enum WeekDays{

MONDAY,

TUESDAY;

}

# Dag 13. Innerclass

public class Container{

private String outsideVar = "Hello from the outside";

//statisk indre klasse, der virker som statisk variabel

// klasse og variabel er defineret "samme sted"

public static class Component{

private int three = 3;

public int getThree(){

return three;

}

}

// en indre klasse

public class InnerClass{

public void printIt(){

//System.out.println(outsideVar);

System.out.println("this");

}

}

// en lokal klasse

// en klasse man laver i en metode

public void createClass(){

class MyLocalClass{

public int million = 1000000;

public void printBucks(){

System.out.println("I have a "+million+" bucks");

}

}

MyLocalClass mlc = new MyLocalClass();

mlc.printBucks();

}

}

public class Run{

//main metode

public static void main(String[] args){

// System.out.println(Container.outsideVar);

//container.getThree();

Container.Component cc = new Container.Component();

System.out.println(cc.getThree());

Container container = new Container();

//container.InnerClass.printIt();

Container.InnerClass ci = container.new InnerClass();

//Container.InnerClass CI = new container.InnerClass();

ci.printIt();

container.createClass();

}

}

Nestede klasser

Stadig Java basic. Men en smule advanced.Typer af nestede klasser

● Statisk indlejret (nested) klasse: Erklæret

som en statisk variabel i en anden klasse

● Indre klasse / inner class: erklæret som en

instans variabel i en anden klasse

● Lokal klasse: Erklæret i en instans metode i en

anden klasse

● Anonym indre klasse: Som lokal klasse, men

skrevet som et udtryk, der returnerer et objektIndre klasse / inner class

public class Rhino {

public class Goat {

}

private void jerry() {

//denne Goat er bundet op på this, en instans af Rhino.

Goat g = new Goat();

}

}

Men udenfor Rhino referer vi til Goat med

Rhino.Goat goat = rhino.new Goat();Static nested class

public class Rhino {

public static class Goat {

}

}

Tilgåes ligesom statiske variabler og metoder.Lokale klasser

● Erklæret i en instans metode i en anden klasse

● Lokale klasser kan ligesom andre nestede

klasser tilgå alle variabler i den klasse, den er

erklæret, inkl private.

● Og den lokale klasse kan tilgå alle variabler etc,

der er i dens eget scope.Anonyme indre klasser

● Anonyme klasser er i virkeligheden subklasser.

● Når man implementerer et interface, så arver man fra den

type som interfacet i virkeligheden er. En klasse er også

en type.

http://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html#

jls-4.12.6

● Så det hedder en anonym klasse fordi den sub-klasse,

man laver ikke har noget navn. Selvom det umiddelbart

ser sådan ud!

● Kan bruges når man vil overskrive én eller to metoder i en

superklasse afhængigt af en anden specifik klasse.

# Dag 14 I\_O exception

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

public class ByteExample{

public static void main(String[] args){

//initialisere vores streams med værdien null

FileInputStream in = null;

FileOutputStream out = null;

try{

in = new FileInputStream("textfil.txt");

out = new FileOutputStream("copy.txt");

int someByte;

while((someByte = in.read()) != -1){

//skrive someBye ud

out.write(someByte);

}

} catch(IOException e){

e.printStackTrace();

} finally{

try{

if(in != null){

System.out.println("It worked :-) !");

in.close();

}

if(out!=null){

out.close();

}

} catch(IOException e){

e.printStackTrace();

}

}

}

}

I dag

● Nestede klasser:

øvelser med

– Static inner classes

– Inner classes

– Local classes

● I/O:

– Streams:

● Byte stream: FileInputStream / FileOutputStream

● Character stream: FileReader / FileWriter

● Buffered stream: BufferedReader / BufferedWriterStatic nested class

public class Rhino {

public static class Goat {

}

}

Tilgåes ligesom statiske variabler og metoder.Lokale klasser

● Erklæret i en instans metode i en anden klasse

● Lokale klasser kan ligesom andre nestede

klasser tilgå alle variabler i den klasse, den er

erklæret, inkl private.

● Og den lokale klasse kan tilgå alle variabler etc,

der er i dens eget scope.Indre klasse / inner class

public class Rhino {

public class Goat {

}

private void jerry() {

//denne Goat er bundet op på this, en instans af Rhino.

Goat g = new Goat();

}

}

Men udenfor Rhino referer vi til Goat med

Rhino.Goat goat = rhino.new Goat();TODO med I/O

● Initialiser en stream.

● Giv streamen et end-point

● Lav en counter af en slags

● Lav et while loop med en metode der returnerer data

fra end-pointed. Kør loopet indtil det er nødvendigt.

● Tag evt det, som metoden returnerer og smid det

over i en outputstream eller hvad du har lyst til.3 ud af 4 nestede klasser

● Static nested klasse: Erklæret som en statisk

variabel i en anden klasse

● Indre klasse: erklæret som en instans variabel

i en anden klasse

● Lokal klasse: Erklæret i en instans metode i en

anden klasseIntroducing I/O

java.lang.Object

(Her kan man jo bare øge størrelsen på Bogstaverne)☺

java.io.Console (implements java.io.Flushable)

java.io.File (implements java.lang.Comparable<T>, java.io.Serializable)

java.io.FileDescriptor

java.io.InputStream (implements java.io.Closeable)

java.io.ByteArrayInputStream

java.io.FileInputStream

java.io.FilterInputStream

java.io.BufferedInputStream

java.io.DataInputStream (implements java.io.DataInput)

java.io.LineNumberInputStream

java.io.PushbackInputStream

java.io.ObjectInputStream (implements java.io.ObjectInput, java.io.ObjectStreamConstants)

java.io.PipedInputStream

java.io.SequenceInputStream

java.io.StringBufferInputStream

java.io.ObjectInputStream.GetField

java.io.ObjectOutputStream.PutField

java.io.ObjectStreamClass (implements java.io.Serializable)

java.io.ObjectStreamField (implements java.lang.Comparable<T>)

java.io.OutputStream (implements java.io.Closeable, java.io.Flushable)

java.io.ByteArrayOutputStream

java.io.FileOutputStream

java.io.FilterOutputStream

java.io.BufferedOutputStream

java.io.DataOutputStream (implements java.io.DataOutput)

java.io.PrintStream (implements java.lang.Appendable, java.io.Closeable)

java.io.ObjectOutputStream (implements java.io.ObjectOutput, java.io.ObjectStreamConstants)

java.io.PipedOutputStream

java.security.Permission (implements java.security.Guard, java.io.Serializable)

java.security.BasicPermission (implements java.io.Serializable)

java.io.SerializablePermission

java.io.FilePermission (implements java.io.Serializable)

java.io.RandomAccessFile (implements java.io.Closeable, java.io.DataInput, java.io.DataOutput)

java.io.Reader (implements java.io.Closeable, java.lang.Readable)

java.io.BufferedReader

java.io.LineNumberReader

java.io.CharArrayReader

java.io.FilterReader

java.io.PushbackReader

java.io.InputStreamReader

java.io.FileReader

java.io.PipedReader

java.io.StringReader

java.io.StreamTokenizer

java.lang.Throwable (implements java.io.Serializable)

java.lang.Error

java.io.IOError

java.lang.Exception

java.io.IOException

java.io.CharConversionException

java.io.EOFException

java.io.FileNotFoundException

java.io.InterruptedIOException

java.io.ObjectStreamException

java.io.InvalidClassException

java.io.InvalidObjectException

java.io.NotActiveException

java.io.NotSerializableException

java.io.OptionalDataException

java.io.StreamCorruptedException

java.io.WriteAbortedException

java.io.SyncFailedException

java.io.UnsupportedEncodingException

java.io.UTFDataFormatException

java.lang.RuntimeException

java.io.UncheckedIOException

java.io.Writer (implements java.lang.Appendable, java.io.Closeable, java.io.Flushable)

java.io.BufferedWriter

java.io.CharArrayWriter

java.io.FilterWriter

java.io.OutputStreamWriter

java.io.FileWriter

java.io.PipedWriter

java.io.PrintWriter

java.io.StringWriter

Interface Hierarchy

java.lang.AutoCloseable

java.io.Closeable

java.io.ObjectInput (also extends java.io.DataInput)

java.io.ObjectOutput (also extends java.io.DataOutput)

java.io.DataInput

java.io.ObjectInput (also extends java.lang.AutoCloseable)

java.io.DataOutput

java.io.ObjectOutput (also extends java.lang.AutoCloseable)

java.io.FileFilter

java.io.FilenameFilter

java.io.Flushable

java.io.ObjectInputValidation

java.io.ObjectStreamConstants

java.io.Serializable

java.io.ExternalizableByte streams

● Byte streams er det mest low-level streams.

● Vi kan selv bruge byte streams.

● Andre streams bruger byte streams

● Byte streams håndterer ét byte ad gangen og

kalder operativ systemet hver gang et byte

læses / skrives.

● FileInputStream og FileOutputStream er

eksempler på byte streams.

# Dag 15 Streams 2

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

public class ByteStream{

// FileInputStream / FileOutputStream

public static void main(String[] args){

FileInputStream in = null;

FileOutputStream out = null;

try{

//prøve at læse fra tekstfil

in = new FileInputStream("text.txt");

out = new FileOutputStream("Byte\_text.txt");

int someByte;

while((someByte = in.read()) != -1){

out.write(someByte);

}

} catch(Exception e){

e.printStackTrace();

} finally {

try{

if(in!= null){

in.close();

}

if(out!= null){

out.close();

}

} catch(IOException e){

e.printStackTrace();

}

}

}

}

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

public class CharacterStream{

// I throw in the main method,

// it will simply terminate if any exception occurs.

public static void main(String[] args) throws IOException{

FileReader inputStream = null;

FileWriter outputStream = null;

try {

inputStream = new FileReader("text.txt");

outputStream = new FileWriter("text\_Character.txt");

int someChar;

while ((someChar = inputStream.read()) != -1) {

outputStream.write(someChar);

}

} finally {

if (inputStream != null) {

inputStream.close();

}

if (outputStream != null) {

outputStream.close();

}

}

}

}

import java.io.FileReader;

import java.io.FileWriter;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.PrintWriter;

import java.io.IOException;

import java.io.OutputStreamWriter;

public class BufferedStream {

public static void main(String[] args) throws IOException {

BufferedReader inputStream = null;

BufferedWriter outputStream = null;

BufferedWriter outputStreamCommand = null;

try {

// her giver vi FileReader ind som parameter til BufferedReader

// så vi gør den underliggende stream til en buffered stream

inputStream = new BufferedReader(new FileReader("tekst.txt"));

// her giver vi FileWriter ind som parameter til BufferedWriter

outputStream = new BufferedWriter(new FileWriter("tekstBuffered.txt"));

outputStreamCommand = new BufferedWriter(new OutputStreamWriter(System.out));

String line;

// here we use the readLine() in the BufferedReader

// to read a whole line at a time

// readLine() returns the read String or null, if we're at the end

while ((line = inputStream.readLine()) != null) {

outputStream.write(line);

outputStream.newLine();

outputStreamCommand.write(line);

outputStreamCommand.newLine();

}

} finally {

if (inputStream != null) {

inputStream.close();

}

if (outputStream != null) {

// close flushes the stream() anyway.

outputStream.close();

outputStreamCommand.close();

}

}

}

}

import java.io.InputStreamReader;

import java.io.FileInputStream;

import java.io.OutputStreamWriter;

import java.io.FileOutputStream;

import java.nio.charset.StandardCharsets; // for at specificere tegnsættet

import java.io.IOException;

public class BridgeStream{

public static void main(String[] args){

// initialisere vores streams med værdien null

//FileReader in = null;

InputStreamReader in = null;

// FileWriter defaulter til OS character encoding.

OutputStreamWriter out = null;

try{

// InputStream is a

in = new InputStreamReader(new FileInputStream("text.txt"), StandardCharsets.UTF\_8);

//File

out = new OutputStreamWriter(new FileOutputStream("text\_Bridge.txt"), StandardCharsets.UTF\_8);

//out = new FileWriter("copyCharacter.txt");

int someByte;

while((someByte = in.read()) != -1){

//skrive someBye ud

out.write(someByte);

}

} catch(IOException e){

e.printStackTrace();

} finally{

try{

if(in != null){

System.out.println("It worked :-) !");

in.close();

}

if(out!=null){

out.close();

}

} catch(IOException e){

e.printStackTrace();

}

}

}

}

// byte streams

import java.io.FileInputStream;

import java.io.FileOutputStream;

// buffered streams

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

// data streams

import java.io.DataInputStream;

import java.io.DataOutputStream;

// exceptions

import java.io.IOException;

import java.io.EOFException;

public class DataStreams {

static final String dataFile = "invoicedata";

static final double[] prices = { 19.95, 9.95, 15.95, 399.95, 100 };

static final int[] units = { 12, 8, 13, 29, 50 };

static final String[] things = { "Computer",

"Cykel",

"Hårtører",

"Regnskov",

"Badekar" };

public static void main(String[] args) throws IOException {

// vi initialiserer vores Data streams

DataOutputStream out = null;

DataInputStream in = null;

// try - catch

try {

// vi lægger end pointet i en byte stream inde i en buffered stream

// inde i en data output stream.

out = new DataOutputStream(new BufferedOutputStream(new FileOutputStream(dataFile)));

// looper igennem vores 3 arrays

// og kalder vores DataOutputStream metoder på hvert element

for (int i = 0; i < prices.length; i ++) {

out.writeDouble(prices[i]);

out.writeInt(units[i]);

out.writeUTF(things[i]);

}

} finally {

out.close();

}

// til at gemme resultatet af det vi har købt

double total = 0.0;

try {

in = new DataInputStream(new

BufferedInputStream(new FileInputStream(dataFile)));

double price;

int unit;

String thing;

try {

while (true) {

price = in.readDouble();

unit = in.readInt();

thing = in.readUTF();

System.out.format("Du har bestilt %d eksemplarer af %s til kr. %.2f%n",

unit, thing, price);

total += unit \* price;

}

} catch (EOFException e) { }

System.out.format("I alt har du brugt: kr. %.2f%n", total);

}

finally {

in.close();

}

}

}

# Dag 16 Strams 3

// arraylist

import java.util.ArrayList;

// byte streams

import java.io.FileInputStream;

import java.io.FileOutputStream;

// buffered streams

import java.io.BufferedInputStream;

import java.io.BufferedOutputStream;

// object streams

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

// exceptions

import java.io.IOException;

import java.io.EOFException;

public class CountryListSerialize{

// the end point of the streams

private static String file = "countries.txt";

public static void main(String[] args){

// making the ArrayList

ArrayList<Country> countryList = new ArrayList<Country>();

// making the objects

Country denmark = new Country("Denmark", 5600000);

Country france = new Country("France", 67000000);

Country norway = new Country("Norway", 5200000);

countryList.add(denmark);

countryList.add(france);

countryList.add(norway);

ObjectInputStream in = null;

ObjectOutputStream out = null;

try{

out = new ObjectOutputStream(new BufferedOutputStream(new FileOutputStream(file)));

out.writeObject(countryList);

//in = new ObjectInputStream(

} catch(Exception e){

e.printStackTrace();

} finally {

try{

out.close();

} catch(IOException e){

e.printStackTrace();

}

}

try{

in = new ObjectInputStream(new BufferedInputStream(new FileInputStream(file)));

ArrayList<Country> countries = (ArrayList<Country>) in.readObject();

for(Country c: countries){

System.out.printf("\nCountry: %s \npopulation: %d\n", c.getName(), c.getPopulation());

}

} catch(Exception e){

e.printStackTrace();

} finally {

try{

in.close();

} catch(IOException e){

e.printStackTrace();

}

}

}

}

// model class to serialize

// interfaces

import java.io.Serializable;

public class Country implements Serializable{

private static final long serialVersionUID = 1L;

private String name;

private int population;

public Country(String name, int population){

this.name = name;

this.population = population;

}

public String getName(){

return name;

}

public int getPopulation(){

return population;

}

}

# Dag 17 java.nio

import java.nio.file.Path;

// utility bibliotek for Path objekter

import java.nio.file.Paths;

// exceptions

import java.nio.file.NoSuchFileException;

import java.io.IOException;

public class PathStuff{

public static void main(String[] args){

// Paths.get("tmp/foo") er short-hand for FileSystems.getDefault().getPath("tmp/foo");

Path path = Paths.get("hej/tmp/foo");

// directory / folder / mappe : tmp/ (unix), tmp\ (windows)

// kan returnere et directory

System.out.println("getFileName: "+ path.getFileName());

System.out.println("getName(0): "+ path.getName(0));

System.out.println("getNameCount: "+path.getNameCount());

System.out.println("getParent: "+path.getParent());

System.out.println("getRoot: "+path.getRoot());

System.out.format("uri :%s\n", path.toUri());

try{

Path realPath = path.toRealPath();

System.out.println(realPath);

} catch(IOException e){

e.printStackTrace();

}

}

}

// paths

import java.nio.file.Path;

import java.nio.file.Paths;

// files

import java.nio.file.Files;

// exceptions

import java.io.IOException;

public class VerifyFile{

public static void main(String[] args){

Path pathToFoo = Paths.get("tmp/foo");

Path pathToBar = Paths.get("tmp/bar");

System.out.println("Foo exists: "+Files.exists(pathToFoo));

// tjekke om filen er et program aka en executable

System.out.println(Files.isExecutable(pathToFoo));

System.out.println(Files.isReadable(pathToFoo));

System.out.println(Files.isWritable(pathToFoo));

try{

Files.delete(pathToFoo);

} catch(Exception e){

e.printStackTrace();

}

System.out.println("Foo exists: "+Files.exists(pathToFoo));

}

}

# Dag 18 Repitation java.nio

// paths

import java.nio.file.Path;

import java.nio.file.Paths;

// files

import java.nio.file.Files;

// exceptions

import java.io.IOException;

import java.nio.file.FileAlreadyExistsException;

import java.io.FileOutputStream;

public class VerifyFile{

public static void main(String[] args){

Path pathToFoo = Paths.get("tmp/foo");

Path pathToBar = Paths.get("tmp/bar");

System.out.println("Foo exists: "+Files.exists(pathToFoo));

// tjekke om filen er et program aka en executable

System.out.println(Files.isExecutable(pathToBar));

System.out.println(Files.isReadable(pathToBar));

System.out.println(Files.isWritable(pathToBar));

try {

Files.createFile(pathToFoo);

} catch(FileAlreadyExistsException e){

System.err.format("file named %s already exists%n", pathToFoo);

} catch(IOException e){

System.err.format("createFile error: %s%n", e);

}

try{

Files.delete(pathToBar);

} catch(Exception e){

e.printStackTrace();

}

//

FileOutputStream out = null;

String s = "oh foo!";

byte[] b = s.getBytes();

try{

out = new FileOutputStream(pathToFoo.toString());

out.write(b);

} catch (IOException e){

}

System.out.println("Foo exists: "+Files.exists(pathToFoo));

}

}

# Dag 19 repitation

## Øvelser med ArrayList

- Lav en ArrayList af Double - i main metoden

- Læg 7-10 doubles i ArrayListen - i main metoden

Kald følgende metoder fra main og skriv det, som metoderne returerer ud:

- Lav en metode der returnerer den højeste Double i listen

- Lav en metode der returnerer den laveste Double i listen

- Lav en metode der returnerer gennemsnittet i listen

- Lav en metode der tager en Double ind og listen som parametrer. Og returnerer true

hvis listen indeholder den Double. Og false hvis den ikke gør.

public abstract class Animal{

// eksempel på composition, Animal has-a Head

protected Head head;

protected int legs = 4;

// abstract method

public abstract void sayHi();

// abstract methods cannot have a body

//NO GOOD: public abstract void sayHi(){}

// non-abstract methods in abstract class are just fine

public void sayGoodBye(){

System.out.println("farewell");

}

public Animal(){}

}

public class Cat extends Animal{

private int legs = 3;

private String color;

private String name;

// this

public Cat(String color){

this(color, "Aladin");// this()

}

public void printNumberOfLegs(){

System.out.println(super.legs);// 4

System.out.println(this.legs); // 3

System.out.println(legs); // 3

}

public Cat(String color, String name){

this.color = color;

this.name = name;

}

public void sayHi(){

System.out.println("Mjaauu");

}

}

public class Head{

private int eyes;

public int getEyes(){

return eyes;

}

public void setEyes(int eyes){

this.eyes = eyes;

}

}

public class Lion extends Animal{

public void setHead(Head head){

this.head = head;

}

protected void sayHi(){

System.out.println("rroooaarrr");

}

}

public class RunClient{

public static void main(String[] args){

//Animal animal = new Animal();

Lion lion = new Lion();

lion.sayHi();

lion.sayGoodBye();

System.out.println("----- END OF LION ------");

System.out.println("\n----- BEGINNING OF CAT ------");

Cat cat = new Cat("Green");

cat.sayHi();

cat.printNumberOfLegs();

System.out.println("----- END OF CAT ------");

}

}

public class DayTest{

public static void main(String[]args){

// enums er smarte fordi vi kender dem allerede

// ved kompilering.

// Og vi kan ikke ændre på værdierne efter kompile

// Og fordi de er eksplicitte omkring hvad de er.

testDay(Days.WEDNESDAY);

printNumberInWeek(Days.FRIDAY);

}

public static void testDay(Days d){

if(d == Days.WEDNESDAY){

System.out.println("Favorite Day");

} else{

System.out.println("not my kind of day");

}

}

public static void printNumberInWeek(Days d){

System.out.println(d.getNumberInWeek());

}

}

public enum Days{

WEDNESDAY(3),

FRIDAY(5),

WEEKEND(6);

private int numberInWeek;

// default constuctor

// public Days(){}

Days(int numberInWeek){

this.numberInWeek = numberInWeek;

}

int getNumberInWeek(){

return numberInWeek;

}

}

public class Day{

public static final String[] days = {"MONDAY", "FRIDAY", "WEEKEND"};

}

# Dag 20 Throw

public class ThrowExample{

public static void main(String[] args){

int yourLuckyNumber = 7;

int myLuckyNumber = 5;

try{

compare(yourLuckyNumber, myLuckyNumber);

} catch(Exception e){

e.printStackTrace();

}

}

private static void compare(int yourLuckyNumber, int myLuckyNumber) throws Exception{

if(myLuckyNumber != yourLuckyNumber){

throw new Exception("We can never be the perfect match");

}

// return yourLuckyNumber == myLuckyNumber;

}

}

public class ReferenceExample{

public static void main(String[] args){

Person p = new Person("Hugo");

System.out.println(p.name);

Person[] persons = new Person[3];

persons[0] = p;

chagePerson(persons);

System.out.println(p.name);

double d = 235.1;

System.out.println(d);

double[] doubles = new double[3];

doubles[0] = d;

changeDoubles(doubles);

System.out.println(d);

}

// metode der tager en reference type ind

public static void chagePerson(Person[] persons){

Person p = persons[0];

p.name = "Joe";

}

public static void changeDoubles(double[] doubles){

double d = doubles[0];

d = 10.0;

}

}

public class Person{

public String name;

public Person(String name){

this.name = name;

}

}

public class Game {

}

public interface Perpetual{

public void continueForever();

}

public class RunClient{

// main

public static void main(String[] args){

Tekken3 t3 = new Tekken3();

t3.win();

Rust rust = new Rust();

if(rust instanceof Perpetual){

rust.continueForever();

}

}

}

public class Rust extends Game implements Perpetual{

public void continueForever(){

System.out.println("impossible to win this game, ha ha ha.");

}

}

public class Tekken3 extends Game implements Winnable{

public void win(){

System.out.println("You beat the master, hence you won");

}

}

public interface Winnable{

public void win();

}

# Dag 20 wrapper class

import java.util.ArrayList;

public class WrapperStuff{

public static void main(String[] args){

double d1 = 4.5; // primitiv type

Double d2 = 6.89; // reference type / Double er en klasse, d2 er et objekt

//double min = Double.MIN\_VALUE; // MIN\_VALUE er en statisk variabel i Double klassen

d2 = d1;

int i = 100;

String s = "Hej";

//s = i;

System.out.println(d2);

// System.out.println(min);

ArrayList<Double> doubles = new ArrayList<Double>();

doubles.add(d1);

for(double d: doubles){

System.out.println(d);

}

}

}