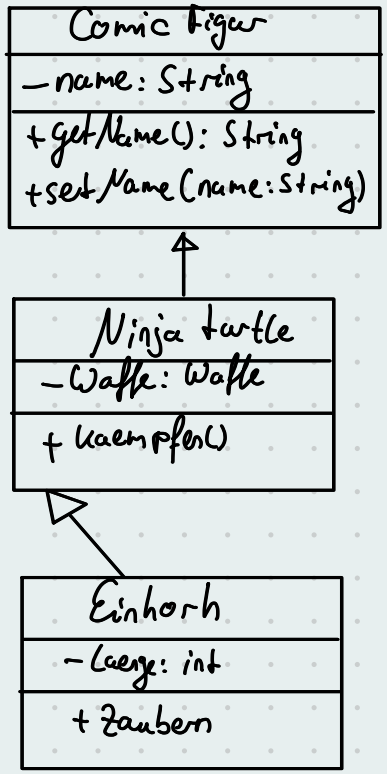
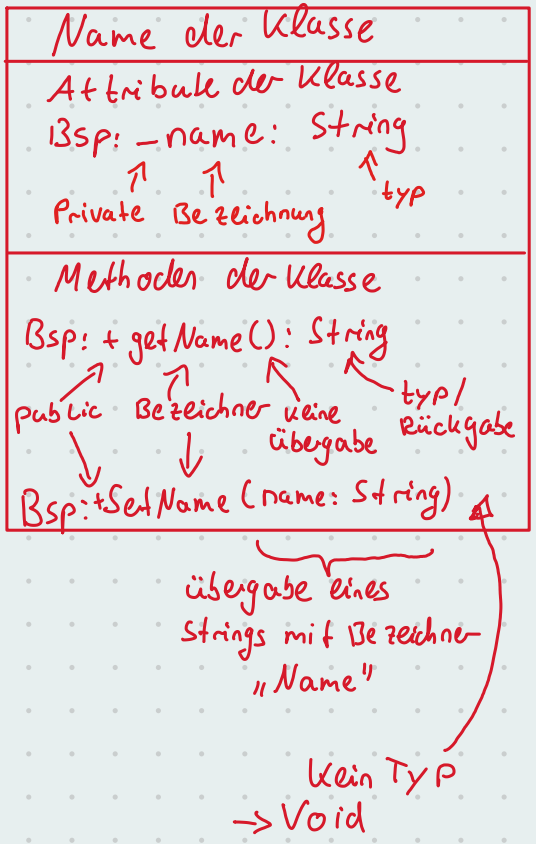
Cheatsheet

UML-Diagramme

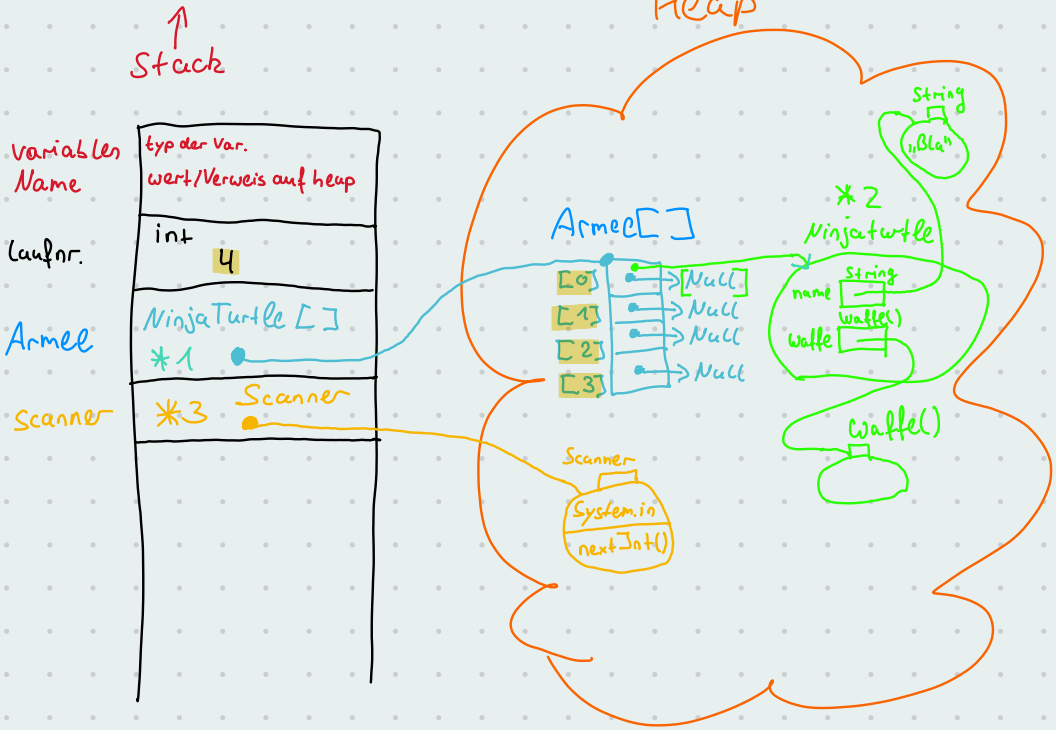




Stack & Heap

Hier werden alle Objekte nach dem Initiieren angezeigt

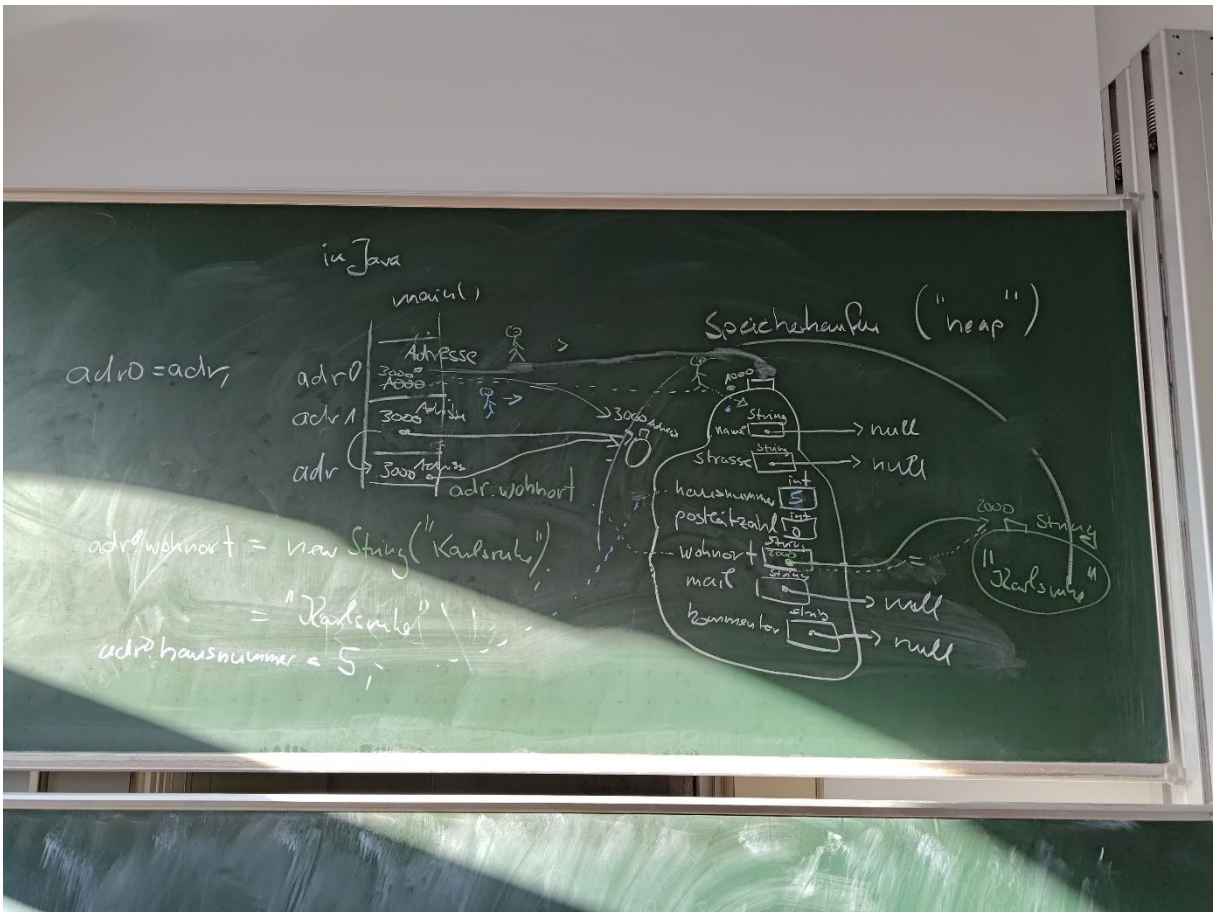
Hier werden die Objekte nach dem Erstellen angezeigt

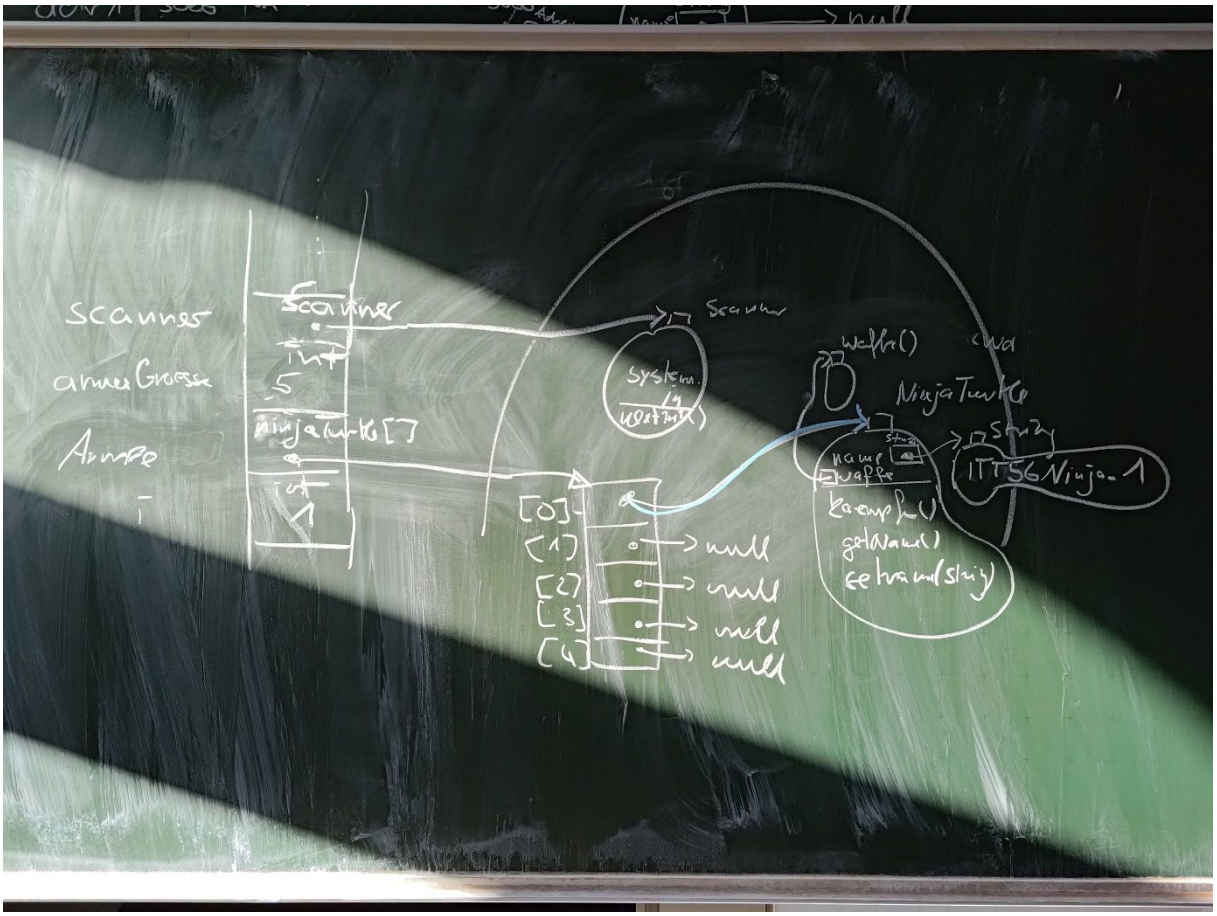


*1: Dieser Zustand tritt auf nach
NinjaTurtle[] Army = new NinjaTurtle[4]
-> Es wird nur platz gemacht

*2: Dieser Zustand überschreibt den Bezug auf Null, und macht einen Bezug zu einem Objekt
Erstmach: Army[i] = new NinjaTurtle;

* nur notwendig wenn man auch Verwendet


in Java
main()
Adresse
adr0 = adr;
adr0
adr1
adr
adr.wohnort
Speicherhaufen ("heap")
String
name
strasse
hausnummer
postleitzahl
wohnort
mail
kommentar
null
null
null
null
adr.wohnort = new String("Karlsruhe");
= "Karlsruhe";
adr.hausnummer = 5;
Karlsruhe


scanner
anzahlGroesse
Anzahl
i
Scanner
int
5
NinjaTurtle[]
int
1
System.in
NinjaTurtle
name
kampf()
getName()
setNameUndString()
ITT56Ninja_1
[0]
[1]
[2]
[3]
[4]
null
null
null
null

CHEAT-SHEET-PROJEKT

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Cheat-Sheet-SA1\src\Elektro.java

```java
public class Elektro extends Fahrzeug {
    // Definition privater Variablen
    private int engineCount;
    private int batterySize;

    // Benutzerdefinierter Konstruktor mit Parametern
    public Elektro(String make, String model, int horsepower, int engineCount, int batterySize) {
        // Parameterloser Vater-Konstruktor -> siehe Konstruktor in Verbrenner.java
        super();

        super.setMake(make);
        super.setModel(model);
        super.setHorsepower(horsepower);

        this.engineCount = engineCount;
        this.batterySize = batterySize;
    }

    // Get-Methoden -> siehe Fahrzeug.java
    public int getEngineCount() {
        return engineCount;
    }

    public int getBatterySize() {
        return batterySize;
    }

    // Set-Methoden -> siehe Fahrzeug.java
    public void setEngineCount(int engineCount) {
        this.engineCount = engineCount;
    }

    public void setBatterySize(int batterySize) {
        this.batterySize = batterySize;
    }

    // Überschreibung einer Methode der Vater-Klasse -> siehe Verbrenner.java
    public void printDataSheet(){
        super.printDataSheet();
        System.out.println("Motorenanzahl: " + engineCount);
        System.out.println("Batteriegröße: " + batterySize);
    }
}

```

CHEAT-SHEET-PROJEKT

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Cheat-Sheet-SA1\src\Katalog.java

```java
// Import für das Anlegen des Scanners
import java.util.Scanner;

public class Katalog {
    public static void main(String[] args) {
        // Einlesen wie groß der Katalog sein soll
        // Scanner-Objekt anlegen -> Scanner vorher importieren!
        Scanner scanner = new Scanner(System.in);
        System.out.println("Wie gross soll der Katalog sein?");
        int catalogSize = scanner.nextInt(); // -> Als Beispiel wurde hier 5 eingegeben
        // Zum Einlesen eines Strings:
        // String inputText = scanner.nextLine();

        // Anlage eines Array, der die vorher definierte Größe hat
        Benziner [] benzinerKatalog = new Benziner[catalogSize];

        // Anlage der Benziner im Array
        benzinerKatalog[0] = new Benziner("BMW", "435i", 340, 3000, 6, "Super-Plus", 98);
        benzinerKatalog[1] = new Benziner("Audi", "S3", 310, 2000, 4, "Super", 95);
        benzinerKatalog[2] = new Benziner("Mercedes-Benz", "C43 AMG", 390, 3000, 6, "
Super-Plus", 98);
        benzinerKatalog[3] = new Benziner("Volkswagen", "Golf GTI", 245, 2000, 4, "Super"
, 95);
        benzinerKatalog[4] = new Benziner("Porsche", "911 Carrera", 385, 3000, 6, "Super-
Plus", 98);

        // Data-Sheet für alle Elemente im Array ausgeben
        for(int i = 0; i < benzinerKatalog.length; i++){
            benzinerKatalog[i].printDataSheet();
        }

        // String Funktionen
        System.out.println("------ String-Funktionen ------");

        String mercedes = benzinerKatalog[2].getMake();
        String lowerCaseMercedes = mercedes.toLowerCase();
        String volkswagen = benzinerKatalog[3].getMake();

        System.out.println(" 1: " + lowerCaseMercedes);
        // "mercedes-benz"
        System.out.println(" 2: " + mercedes.charAt(3));
        // "c" -> Der Character an der 4. Position
        System.out.println(" 3: " + mercedes.compareTo(volkswagen));
        // "-9" -> "V" olkswagen ist 9 Stellen unter "M" ercedes-Benz
        System.out.println(" 4: " + volkswagen.endsWith("wagen"));
        // "true"
        System.out.println(" 5: " + mercedes.equals(lowerCaseMercedes));
        // "false" -> da Case-Sensitive
        System.out.println(" 6: " + mercedes.equalsIgnoreCase(lowerCaseMercedes));
        // "true" -> nicht Case-Sensitive
        System.out.println(" 7: " + mercedes.indexOf("-"));
        // "8" -> Bindestrich an 8. Stelle
        System.out.println(" 8: " + mercedes.indexOf("Ben"));
        // "9" -> beginn des substrings an 9. Stelle
        System.out.println(" 9: " + mercedes.indexOf("BMW"));
        // "-1" -> Substring ist nicht in String enthalten
        System.out.println("10: " + mercedes.length());
        // "13" -> "Mercedes-Benz" hat 13 Zeichen
        System.out.println("11: " + mercedes.replace('e', 'x'));
        // "Mxrcxdxs-Bxnz" -> alle 'e' werden durch 'x' ersetzt
        System.out.println("12: " + mercedes.repeat(2));
        // "Mercedes-BenzMercedes-Benz" -> String wird zwei mal hintereinander gehängt
        System.out.println("13: " + mercedes.startsWith("Schoen"));
        // "false" -> String beginnt nicht mit "Schoen"
        System.out.println("14: " + mercedes.substring(4));
        // "edes-Benz" -> String beginnt an 4. Stelle
        System.out.println("15: " + mercedes.substring(5,12));
```

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Cheat-Sheet-SA1\src\Katalog.java

```
65          // "des-Ben" -> string beginnt an 5. Stelle und endet VOR 12.
66          System.out.println("16: " + volkswagen.toLowerCase());
67          // "volkswagen" -> alle Großbuchstaben werden in Kleinbuchstaben umgewandelt
68          System.out.println("17: " + volkswagen.toUpperCase());
69          // "VOLKSWAGEN" -> alle Kleinbuchstaben werden in Großbuchstaben umgewandelt
70          System.out.println("18: " + String.valueOf(1.5e2));
71          // "150.0" -> wird in lesbaren String umgewandelt
72
73
74          // Datentypen in JAVA
75
76          // Primitive Datentypen
77          // -> Werden im Speicher direkt in der Zelle gespeichert
78          boolean isCar = true;       // Wahr/Falsch-Wert
79          char character = 'a';       // Ein einzelnes Zeichen
80          int number = 0;             // Ganze Zahl
81          float floatNumber = 1.5F;   // Kommazahl mit einfacher Genauigkeit
82          double doubleNumber = 1.5D; // Kommazahl mit doppelter Genauigkeit
83
84          byte smallNumber = 127;     // Kleine Ganzzahl (-128 bis 127)
85          short mediumNumber = 32000; // Mittelgroße Ganzzahl (-32,768 bis 32,767)
86          long largeNumber = 123456789L; // Große Ganzzahl (bis ±9 Quintillionen)
87
88          // Non-Primitive/Komplexe Datentypen
89          // -> Speicherzelle enthält einen Pointer auf ein Objekt im Heap sobald es
    erstellt wurde.
90          // Vor der Initialisierung enthält die Speicherzelle einen Pointer auf null
91          int [] numbers = new int[10]; // Array jeglichen Datentyps
92          Fahrzeug testFahrzeug = new Fahrzeug(); // Jede Klasse ist ein komplexer
    Datentyp
93          String text = "Das ist ein String"; // Ein String ist ein Objekt
94      }
95  }
96
```

CHEAT-SHEET-PROJEKT

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Cheat-Sheet-SA1\src\Benziner.java

```java
public class Benziner extends Verbrenner  {
    // Definition privater Variablen
    private String fuelSpecification;
    private int octaneRating;

    // Benutzerdefinierter Konstruktor mit Parametern -> siehe Verbrenner.java
    public Benziner(String make, String model, int horsepower, int displacement, int cylinderCount, String fuelSpecification, int octaneRating) {
        // Es wird nun der Konstruktor der Vater-Klasse "Verbrenner" aufgerufen. In dessen Konstruktor werden allerdings
        // auch Attribute der, aus aktueller Sicht, Großvaterklasse "Fahrzeug" gesetzt. Daher benötigt der Konstruktor
        // der Klasse "Benziner" als Parameter sowohl die Werte für die Attribute der Klasse "Verbrenner" als auch der Klasse "Fahrzeug"
        super(make, model, horsepower, displacement, cylinderCount);

        // Da ein Benzinmotor immer Benzin als Treibstoff braucht, kann das entsprechende Attribut der Vater-Klasse "Verbrenner"
        // im Konstruktor gesetzt werden
        super.setFuelType("Benzin");

        this.fuelSpecification = fuelSpecification;
        this.octaneRating = octaneRating;
    }

    // Get-Methoden -> siehe Fahrzeug.java
    public String getFuelSpecification() {
        return fuelSpecification;
    }

    public int getOctaneRating() {
        return octaneRating;
    }

    // Set-Methoden -> siehe Fahrzeug.java
    public void setFuelSpecification(String fuelSpecification) {
        this.fuelSpecification = fuelSpecification;
    }

    public void setOctaneRating(int octaneRating) {
        this.octaneRating = octaneRating;
    }

    // Erneutes Überschreiben einer Methode der Vater-Klasse: Die Methode "printDataSheet" wurde in der Klasse "Fahrzeug" definiert
    // und anschließend in "Verbrenner" überschrieben. Beim Überschreiben wurde allerdings die Methode der Vater-Klasse mit verwendet
    // um nicht unnötig Code zu duplizieren. Das Gleiche kann in der Tochter-Klasse "Benziner" der Tochter-Klasse "Verbrenner" gemacht
    // werden. Dabei wird nicht die Methode der Klasse "Fahrzeug" überschrieben, sondern die Methode der Klasse "Verbrenner". Beim
    // Überschreiben kann mittels "super" die überschriebene Methode aus "Verbrenner" verwendet werden.
    public void printDataSheet(){
        // Aufruf der Methode aus der Vater-Klasse "Verbrenner"
        super.printDataSheet();
        System.out.println("Kraftstoff Spezifikation: " + fuelSpecification);
        System.out.println("Oktanzahl : " + octaneRating);
    }
}

```

CHEAT-SHEET-PROJEKT

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Cheat-Sheet-SA1\src\Fahrzeug.java

```java
public class Fahrzeug {
    // private Variablen erstellen
    private String make;
    private String model;
    private int horsepower;

    // An dieser Stelle könnte ein Konstruktor stehen. Der Konstruktor wird ausgeführt, sobald eine neue Instanz eines
    // Objekts erstellt wird. Wenn kein Konstruktor definiert ist, gibt es einen parameterlosen Konstruktor. Das Objekt
    // wird also initialisiert, aber es passiert nichts weiter.

    // Da private Variablen nicht per Punktreferenz (Objekt.attribut) werden Get-Methoden benötigt,
    // die public sind. Dadurch kann ein privater Parameter trotzdem per Punktreferenz abgerufen werden
    // (Objekt.getAttribut())
    public String getMake() {
        return make;
    }

    public String getModel() {
        return model;
    }

    public int getHorsepower() {
        return horsepower;
    }

    // Set-Methoden sind das Gegenstück von Get-Methoden. Damit lassen sich private Variablen außerhalb des
    // Objekts setzen
    public void setMake(String make) {
        this.make = make;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public void setHorsepower(int horsepower) {
        this.horsepower = horsepower;
    }

    // Weitere Methoden
    public void printDataSheet(){
        System.out.println("----- Fahrzeug-Info-Blatt -----");
        System.out.println("Marke: " + make);
        System.out.println("Modell: " + model);
        System.out.println("Leistung: " + horsepower);
    }
}

```

CHEAT-SHEET-PROJEKT

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Cheat-Sheet-SA1\src\Verbrenner.java

```java
public class Verbrenner extends Fahrzeug{
    // Definition von privaten Variablen
    private int displacement;
    private int cylinderCount;
    private String fuelType;

    // Konstruktor der Verbrenner-Klasse
    public Verbrenner(String make, String model, int horsepower, int displacement, int cylinderCount) {
        // Zuerst muss IMMER der Konstruktor der Vater-Klasse ausgeführt werden! Ein parameterloser Konstruktor muss
        // nicht zwingend genannt werden. Er wird automatisch aufgerufen, sollte nichts anderes in der Tochter-Klasse
        // definiert sein,
        super();

        // Da der Konstruktor der Tochter-Klasse "Verbrenner" auch Parameter annimmt, die der Vater-Klasse zugehörig
        // sind, sollten diese auch entsprechend gesetzt werden. Da die Variablen der Vater-Klasse allerdings private
        // sind, können sie nicht direkt (super.make = make) gesetzt werden, sondern müssen über die Set-Methoden
        // gesetzt werden
        super.setMake(make);
        super.setModel(model);
        super.setHorsepower(horsepower);

        // Anschließend werden die privaten Variablen der Tochter-Klasse gesetzt
        this.displacement = displacement;
        this.cylinderCount = cylinderCount;
    }

    // Get-Methoden → siehe Fahrzeug.java
    public int getDisplacement() {
        return displacement;
    }

    public int getCylinderCount() {
        return cylinderCount;
    }

    public String getFuelType() {
        return fuelType;
    }

    public int getDisplacementPerCylinder() {
        return displacement / cylinderCount;
    }

    // Set-Methoden → siehe Fahrzeug.java
    public void setDisplacement(int displacement) {
        this.displacement = displacement;
    }

    public void setCylinderCount(int cylinderCount) {
        this.cylinderCount = cylinderCount;
    }

    public void setFuelType(String fuelType) {
        this.fuelType = fuelType;
    }

    // Überschreibung einer Methode der Vater-Klasse:
    // Wenn ein Vater-Objekt eine Methode besitzt, die in einer Tochter-Klasse anders behandelt werden muss,
    // kann diese überschrieben werden. Dazu muss eine Methode mit dem gleichen Methoden-Kopf erstellt werden.
```

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Cheat-Sheet-SA1\src\Verbrenner.java

```
60
61      public void printDataSheet(){
62          // Innerhalb einer Methode kann die Methode des Vater-Objekts mit super.<Methode
    >() aufgerufen werden.
63          // Das beschränkt sich nicht auf überschriebene Methoden. In diesem Fall macht
    das allerdings hier Sinn:
64          // Die Methode erweitert das Basis-Data-Sheet aus dem Vater-Objekt um
    Informationen aus dem Tochter-Objekt.
65
66          // Ohne den Super-Aufruf müsste der gesamte Inhalt der gleichnamigen Methode in
    Fahrzeug.java eingefügt werden.
67          // Man spart sich also Code-Zeilen
68          super.printDataSheet();
69          System.out.println("Hubraum: "+ displacement + " ccm");
70          System.out.println("Zylinder: "+ cylinderCount + " (" +
    getDisplacementPerCylinder() + " ccm)");
71          System.out.println("Kraftstoff: "+ fuelType);
72      }
73  }
74
```

TRAININGSLAGER SA1

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Trainingslager-SA1\src\Waffe.java

```java
public class Waffe {
    public String name;
}

```

TRAININGSLAGER SA1

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Trainingslager-SA1\src\Einhorn.java

```java
public class Einhorn extends ComicFigur {
    public void zaubern(){
        System.out.println(this.getName() + " zaubert jetzt!");
    }
}

```

TRAININGSLAGER SA1

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Trainingslager-SA1\src\NinjaTest.java

```java
public class NinjaTest {
    public static void main(String[] args) {
        int amount = -1;
        String baseName = "ITT56Ninja_";

        // read count from console
        System.out.print("Anzahl der NinjaTurtles: ");
        amount = Integer.parseInt(System.console().readLine());

        NinjaTurtle army [] = new NinjaTurtle[amount];

        for(int i = 0; i<amount; i++){
            army[i] = new NinjaTurtle();
            army[i].setName(baseName + i+1);
            army[i].kaempfen();
            System.out.println(army[i].getName());
        }
    }
}

```

TRAININGSLAGER SA1

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Trainingslager-SA1\src\ComicFigur.java

```java
public class ComicFigur {
    private String name;

    public String getName(){
        return this.name;
    }

    public void setName(String name){
        this.name = name;
    }
}

```

TRAININGSLAGER SA1

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Trainingslager-SA1\src\NinjaTurtle.java

```java
public class NinjaTurtle extends ComicFigur{
    private Waffe waffe;

    public void kaempfen(){
        System.out.println(this.getName() + " beginnt zu kämpfen");
    }
}

```

VIDEOBAND

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Videoband\src\Reportage.java

```java
public class Reportage extends Videoband {
    String releaseDate;

    public Reportage(String ttl, String releaseDate) {
        super(ttl);
        this.releaseDate = releaseDate;
    }

    public Reportage(String ttl, int duration, String releaseDate){
        super(ttl, duration);
        this.releaseDate = releaseDate;
    }

    public void anzeigen(){
        super.anzeigen();
        System.out.println("Release: " + releaseDate);
    }

}

```

VIDEOBAND

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Videoband\src\Spielfilm.java

```java
public class Spielfilm extends Videoband
{
    String  regisseur;     // Name des Regisseurs
    String  bewertung;     // G, PG, R, oder X

    // Konstruktor
    public Spielfilm( String ttl, int len, String reg, String bew )
    {
        super( ttl, len ); // den Konstruktor der Superklasse verwenden
        regisseur = reg;   // initialisieren, was in Spielfilm neu ist
        bewertung = bew;
    }

    public Spielfilm( String ttl, String reg, String bew )
    {
        super( ttl ); // den Konstruktor der Superklasse verwenden
        regisseur = reg;   // initialisieren, was in Spielfilm neu ist
        bewertung = bew;
    }

    public void anzeigen(){
        System.out.println( titel + ", " + laenge + " Min. verfuegbar: " + vorhanden +
" - Regisseur: " + regisseur + " - Bewertung: " + bewertung );
    }

}
```

VIDEOBAND

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Videoband\src\Videoband.java

```java
public class Videoband
{
    String   titel;        // Titel des Videos
    int      laenge;       // Anzahl der Minuten
    boolean  vorhanden;    // ist das Video vorhanden?

    // Konstruktor
    public Videoband( String ttl )
    {
        titel = ttl; laenge = 90; vorhanden = true;
    }

    // Konstruktor
    public Videoband( String ttl, int len )
    {
        titel = ttl; laenge = len; vorhanden = true;
    }

    public void anzeigen()
    {
        System.out.println( titel + ", " + laenge + " Min. verfuegbar: " + vorhanden );
    }

}
```

VIDEOBAND

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Videoband\src\Musikvideo.java

```java
public class Musikvideo extends Videoband
{
    String kuenstler;
    String kategorie;

    // Konstruktor
    public Musikvideo ( String ttl, int len, String kuenst, String kat )
    {
        super( ttl, len );
        kuenstler = kuenst;
        kategorie = kat;
    }

    public void anzeigen()
    {
        super.anzeigen();
        System.out.println( "Kuenstler:" + kuenstler + " Kategorie: " + kategorie );
    }

}
```

VIDEOBAND

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Videoband\src\VideoVerleih.java

```java
public class VideoVerleih
{
    public static void main ( String args[] )
    {
        TechnikReportage tReportagen [] = new TechnikReportage[5];
        tReportagen[0] = new TechnikReportage("Der Fall Boeing", 70, "10.06.2014", "Flugzeuge");
        tReportagen[1] = new TechnikReportage("Der Tesla-Mythos", 85, "22.08.2017", "Elektroautos");
        tReportagen[2] = new TechnikReportage("Das Geheimnis der Quantencomputer", 95, "15.11.2020", "Computer");
        tReportagen[3] = new TechnikReportage("Die Wiedergeburt der Raumfahrt", 78, "04.07.2019", "Raumfahrt");
        tReportagen[4] = new TechnikReportage("Künstliche Intelligenz: Fluch oder Segen?", 90, "12.03.2022", "KI-Technologie");

        for (TechnikReportage tRep : tReportagen) {
            tRep.anzeigen();
        }
    }
}
```

VIDEOBAND

File - J:\Repositories\IT-Techniker-Erlangen\24-25\PRG\Videoband\src\TechnikReportage.java

```java
public class TechnikReportage extends Reportage {
    String topicArea;

    public TechnikReportage(String ttl, String releaseDate, String topicArea) {
        super(ttl, releaseDate);
        this.topicArea = topicArea;
    }

    public TechnikReportage(String ttl, int duration, String releaseDate, String topicArea) {
        super(ttl, duration, releaseDate);
        this.topicArea = topicArea;
    }

    public void anzeigen(){
        super.anzeigen();
        System.out.println("Topic Area: " + topicArea);
    }
}

```