Marvin Glaser
4424114

Gruppe 9
Wilhelm Esser

Programmieren 1 (PRG1) Übung 6

6.2

Konstruktor:

Eine Methode die zur Definition von Klassenstrukturen in der OOP genutzt wird. Der Konstruktor wird einmal ausgeführt, sobald die Klassenstruktur (Definition der Klasse) zum ersten Mal durchgeführt wird. Mithilfe des Konstruktors können einer Instance der Klasse, bei ihrer Definition, benutzerdefinierte oder festgelegte Werte zugewiesen werden.

Quelle: https://www.techopedia.com/definition/5656/constructor; Datum: 30.11.17; Uhrzeit: 21:50

Destruktor:

Eine Methode die Ausgefürht wird, sobald ein Objekt für das sie definiert wurde zerstört wird. Die Methode wird häufig dazu verwendet übrige oder ungewollte „Datenreste“ zu entfernen, Verbindungen zu Servern zu beenden o.ä. („clean up“). Beim programmieren von Klassen, werden Destruktoren dazu verwendet eine bestehende Klassenstruktur aufzulösen.

Quelle: https://www.techopedia.com/definition/24284/destructor; Datum: 30.11.17; Uhrzeit: 22:00;

https://en.wikipedia.org/wiki/Object_lifetime; Datum: 30.11.17; Uhrzeit: 22:00

public:

Auch andere Klassen können Attribute/Methoden die als „public“ definiert sind, sowohl sehen, als auch modifizieren.

private:

Nur die eigene Klasse kann Änderungen an den Attributen/Methoden der Klasse durchführen.

Protected:

Nur die eigne Klasse und Subklassen von der „geschützten“ Klasse, können die Attribute/Methoden der „protected“ class modifizieren.

Quellen: Vorlesungsfolien: „V12-ObjektorientierteProgrammierung-Klassen“; Datum: 30.11.17; Folien 22,61,62; Vorlesungsskript: „V10-Klassen“; Datum: 30.11.17; Seiten 3-4 („Vorteile der Kapselung“)

https://stackoverflow.com/questions/1020749/what-are-public-private-and-protected-in-object-oriented-programming; Datum: 30.11.17; Uhrzeit: 22:30

Marvin Glaser
4424114

Gruppe 9
Wilhelm Esser

6.3.1

a)

Output:

```
Bike number 1 is a Motorcycle.
The bike has driven a total of 0 Kilometers.
The motor of the bike has 100 hp.
```

Ausgaben in gegebenem Beispiel sind die Attribute vehicel (bike1.vehicle), Kilometerstand (bike1.mileage) und Pferdestärken (bike1.horsepower) der Instance bike1.

Der Parameter vehicle wurde gewählt, um die Art des Fahrzeugs darzustellen. Die Pferdestärke war einfach eine Typische Angabe von Fahrzeuginformationen.

b)

Output 1:

```
Color scheme:
The color of bike1 is "Sarcoline".
The color of bike2 is "Smaragdine".
```

Print der in main() erzeugten Attribute bike1.color und bike2.color. Die Attribute wurden mittels bike1.set_color(„x“) (bzw. bike2.set set_color(„x“)) erzeugt. Die untypischen Farben „Sarcoline“ (heißt so viel wie „Fleischfarben“), bzw. „Smaragdine“ (Smaragdgrün) wurden zur Erheiterung des Lesers gewählt.

Siehe nächste Seite

Output 2:

```
Test drive, Round 1:
During round 1 bike1 drove 128 km,
while bike2 drove 146 km.

The total mileage of bike1 is now: 128 km.
The total mileage of bike2 is now: 146 km.


Test drive, Round 2:
During round 2 bike1 drove 143 km,
while bike2 drove 284 km.

The total mileage of bike1 is now: 271 km.
The total mileage of bike2 is now: 430 km.


Test drive, Round 3:
During round 3 bike1 drove 186 km,
while bike2 drove 476 km.

The total mileage of bike1 is now: 457 km.
The total mileage of bike2 is now: 906 km.
```

Mittels .randrange() Funktion, wurden in der Funktion test_drive(bike1, bike2, 3) über drei Runden hinweg jeweils zwei Zahlen zwischen 100 und 500 ermittelst. Das bike.mileage() Attribut jedes Bikes wurde um den entsprechenden Wert angehoben und die Ergebnisse jeder Runde in der Konsole ausgedruckt.

6.3.2

c)

Output:

```
bike1:
The maximum reach of this vehicle are 25.0 km.
```

Mittels Konstruktor wurden die Attribute curr_capacity und consumption bei der Definition von bike1 festgelegt. Im Anschluss wurde Funktion show_reach() ausgeführt. Diese berechnet das Verhältnis von Kapazität und Verbrauch und druckt das Ergebnis in der Konsole aus.

d)

Output:

```
ebike1:
The maximum reach of this vehicle are 25.0 km.

ebike2
The maximum reach of this vehicle are 15.0 km.
```

Die Eingabe erfolgte analog zu Aufgabe c).

Output help(Ebike):

```
Help on class Ebike in module __main__:

class Ebike(bike)
 |  New class "bike"
 |  new class contains four attributes and three methods
 |
 |  Method resolution order:
 |      Ebike
 |      bike
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, curr_capacity, consumption)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  show_reach(self)
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from bike:
 |
 |  inc_miles(self, kilometers)
 |      this function adds a defined integer to the mileage attribute
 |
 |  set_color(self, color)
 |      this function creates/redifines the color attribute of a bike
 |      instance.
 |
 |  show_color(self)
 |      this function prints the color attribute of the instance
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from bike:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes inherited from bike:
 |
 |  horsepower = 100
 |
 |  mileage = 0
 |
 |  vehicle = 'motorcycle'
```

Die Funktion help() zeigt, dass Ebike eine Subklasse von Bike ist. Daher wurden zuerst alle in Bike definierten Funktionen und dann alle Attribute von Bike übernommen.