Einführung in die imperative Programmierung in Java Teil 2

Regionalzentrum Hamburg Michael Zöller

Agenda

- Wiederholung
- Klassen und Felder
- Methoden
- Objekte
- Enums
- Interface
- Vererbung

FizzBuzz

Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”. Separate with comma and print everything in one line. Don’t add a comma at the end.

```
1, 2, Fizz, 4, Buzz, Fizz, 7, […], 98, Fizz, Buzz
```

Agenda

Wiederholung

Klassen und Felder

Methoden

Objekte

Enums

Interface

Vererbung

Klassen 1

- Packages
- Access modifier
- Felder
- Methoden

Felder

- Access modifier
- Deklaration vs. Initialisierung
- Blöcke
- Konstanten
- Final
- static

Klassenfelder/-methoden: static

- Static fields: Gültig für alle Objekte der Klasse (globale Variable, shared state!!!)
- Static methods: Zum Zugriff auf static fields, Zugriff über Klasse

Agenda

Wiederholung

Klassen und Felder

Methoden

Objekte

Enums

Interface

Vererbung

Methoden

- Signatur
- Formal- und Aktualparameterliste
- Rückgabetyp, void
- Copy by value, copy by reference, copy by sharing (call by value where the value is a reference)

Call by reference/ call by value, immutability

- Was übergibt man bei einem Methodenaufruf
  - Einen Alias des Übergabeparameters (call by reference)
  - Eine Kopie des Übergabeparameters (call by value)?
- In Java immer eine Kopie (call by value). ABER: das ist nur bei primitiven Datentype intuitiv, da bei Objekten eine Kopie des Zugriffs auf den Objektinhalt übergeben wird (Adresse, Zeiger, …)
- Das bedeutet, dass man zwar den Übergabewert nicht ändern kann, wohl aber dessen „Inhalt“ (genauer Begriff hier: call by sharing)

Agenda

Wiederholung

Klassen und Felder

Methoden

Objekte

Enums

Interface

Vererbung

Klasse 2: Objekte

- Konstruktoren
- New

Agenda

Wiederholung

Klassen und Felder

Methoden

Objekte

Enums

Interface

Vererbung

Enum

- Aufzählungstypen
  - Z.B. Wochentag (Montag, Dienstag,…,Sonntag)
- Neu in Java 5, haben Konstantenklassen/-interfaces abgelöst
- Jede Ausprägung ist ein Singleton (vergleichbar also mit ==)

Agenda

Wiederholung

Klassen und Felder

Methoden

Objekte

Enums

Interface

Vererbung

Interfaces

- Beschreiben einen Typ, oder auch eine Rolle bzw. eine Schnittstelle
- Erhält Methodensignaturen
- Kann ausimplementierte static bzw. default Methoden enthalten
- Kann Konstanten enthalten
- Klassen können Interfaces implementieren

Polymorphie

- Meint Vielgestaltigkeit
- Nutzt Typ-Vererbung und „late binding“ aus
- Late Binding: erst zur Laufzeit wird die konkrete Implementierung einer Methode ermittelt

Annotation @Override

- Mit Annotation werden Metadaten und Steuerungsinformationen hinterlegt (seit Java 5)
- Vermutlich eines der wichtigsten und mächtigsten Features, die jemals Java hinzugefügt worden sind
- Die @Override Annotation ist eher harmlos. Damit weißt man eben darauf hin, dass hier eine Methode überschrieben werden soll
- Falls mit der Signatur irgendetwas nicht passt, kann der Compiler meckern

Agenda

Wiederholung

Klassen und Felder

Methoden

Objekte

Enums

Interface

Vererbung

Vererbung

- Typ-Vererbung: eine **IST-EIN** Beziehung (implements Interface) [subtyping]
- Implementierungs-Vererbung: eine **IST-EIN** Beziehung und erben von öffentlichen und protected Feldern und Methoden (extends Superclass) [subclassing]
- In Java ist Einfach-Subclassing und Mehrfach-Subtyping möglich

Overwriting vs. overloading und Object

- Subklassen können Methoden ihrer Superklassen überschreiben (toString())
- Alle Java-Klassen erben implizit von Objekt
- Besondere Relevanz neben toString() haben die Methoden equals() und hashcode(), die immer gemeinsam überschrieben werden müssen
- Dagegen meint overloading, dass es Methoden gleichen Namens mit unterschiedlicher Formalparameterliste gibt

Delegation

- Implementierungs-Vererbung gibt es in Java nur zusammen mit Typ-Vererbung – das hat durchaus Nachteile
- Möchte man nur Code an einer Stelle bündeln, setzt man besser auf Delegation: eine **HAT-EIN** Beziehung

Links

- https://scratch.mit.edu/
- http://www.tutego.de/javabuch/index.html
- https://docs.oracle.com/javase/tutorial/
- https://stackoverflow.com/
- https://www.amazon.de/Core-Java-I-Fundamentals-Cay-Horstmann/dp/0134177304/ref=sr_1_1