

## Übung HTTP Code Map

### Ausgangslage

Bei der Übung Switch wurde jedem HTTP Code die richtige Fehlermeldung zugewiesen. Diese Zuordnung wurde mit einem switch-Statement gelöst. Mit dieser Übung soll die gleiche Anwendung nun über eine HashMap in Form der Klasse `java.util.HashMap` gelöst werden. Hierzu ist als Key-Objekt der HTTP-Code zu verwenden. Das Value-Objekt soll dem zugeordneten Text entsprechen. Das HTTP-Protokoll definiert die folgenden Successful-Codes mit Text: 100 Continue 200 OK 201 Created 202 Accepted 203 Non-Authoritative Information 204 No Content 205 Reset Content 206 Partial Content. Das HTTP-Protokoll definiert die folgenden Redirection-Codes mit Text: 300 Multiple Choices 301 Moved Permanently 302 Moved Temporarily 303 See Other 304 Not Modified 305 Use Proxy. Das HTTP-Protokoll definiert die folgenden Client-Error-Codes mit Text: 400 Bad Request 401 Unauthorized 402 Payment Required 403 Forbidden 404 Not Found 405 Method Not Allowed 406 Not Acceptable 407 Proxy Authentication Required 408 Request Timeout 409 Conflict 410 Gone 411 Length Required 412 Precondition Failed 413 Request Entity Too Large 414 Request-URI Too Long 415 Unsupported Media Type. Das HTTP-Protokoll definiert die folgenden Server-Error-Codes mit Text: 500 Internal Server Error 501 Not Implemented 502 Bad Gateway 503 Service Unavailable 504 Gateway Timeout 505 HTTP Version Not Supported. Kommandozeilenparameter werden beim Aufruf dem Programm übergeben. Die Parameter werden mit einem Space (Blank) separiert und im Array `args[]` der Methode `main(...)` abgelegt. Das folgende Codefragment zeigt die Auswertung und Anzeige von Kommandozeilenparametern:

```
public static void main (String[] args) {
    for (int i=0; i < args.length; i++) {
        System.out.println ("#34;param#34; + i + #34;: #34; + args[i]);
    }
}
```

Beim Ausdruck `args[i]` handelt es sich um einen String. Dieser String ist nun in einen primitiven int-Datentyp zu konvertieren. Das folgende Codefragment zeigt diese Konversion über die Wrapper-Klasse `Integer` auf:

```
int number = Integer.parseInt (args[i]);
```

### Vorgehen

- Erzeugen Sie ein neues Java Projekt oder ein Package für diese Übung.
- Erzeugen Sie die Klasse `HttpCodeMap`.
- Programmieren Sie die Klasse `HttpCodeMap` mit dem Gerüst der Methode `main(String []args)` aus.
- Werten Sie den Kommandozeilen-Parameter aus, der den Fehlercode beinhaltet.  
Hinweis: Überprüfen Sie auch den Fall, dass kein Parameter vorhanden ist.
- Instanziiieren Sie nun ein Objekt vom Type `java.util.HashMap` und fügen Sie alle Http Codes in die Table ein. Das Key Objekt soll dem Http Code (Wrapper Klasse `java.lang.Integer`) und das Value-Objekt dem Text entsprechen.
- Holen Sie nun zum gegebenen Http Code den entsprechenden Text aus der HashMap und zeigen Sie diesen an.
- Falls der Http Code nicht existiert, so soll dies entsprechend mitgeteilt werden.

### Zusatzaufgabe Print Entries

- Geben Sie alle Map Entries mit der for-each Schleife aus.
- Geben Sie alle Map Entries mit der forEach Lambda Funktion aus.

### Zusatzaufgabe Tree Map

- Kopieren Sie die Lösung um in die Klasse HttpStatusCodeTreeMap und ersetzen Sie die HashMap mit der TreeMap.

- Was stellen Sie fest.

### Lösung

Eine mögliche Lösung finden Sie hier

#### Kontakt

Simtech AG  
Finkenweg 23  
3110 Münsingen  
Schweiz

#### Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.  
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsausschluss.  
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

#### Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

<https://www.simtech-ag.ch/kurse-jpf2-http-code-map>