//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package org.springframework.http;  
  
import org.springframework.lang.Nullable;  
  
public enum HttpStatus {  
 *CONTINUE*(100, HttpStatus.Series.*INFORMATIONAL*, "Continue"),  
 *SWITCHING\_PROTOCOLS*(101, HttpStatus.Series.*INFORMATIONAL*, "Switching Protocols"),  
 *PROCESSING*(102, HttpStatus.Series.*INFORMATIONAL*, "Processing"),  
 *CHECKPOINT*(103, HttpStatus.Series.*INFORMATIONAL*, "Checkpoint"),  
 *OK*(200, HttpStatus.Series.*SUCCESSFUL*, "OK"),  
 *CREATED*(201, HttpStatus.Series.*SUCCESSFUL*, "Created"),  
 *ACCEPTED*(202, HttpStatus.Series.*SUCCESSFUL*, "Accepted"),  
 *NON\_AUTHORITATIVE\_INFORMATION*(203, HttpStatus.Series.*SUCCESSFUL*, "Non-Authoritative Information"),  
 *NO\_CONTENT*(204, HttpStatus.Series.*SUCCESSFUL*, "No Content"),  
 *RESET\_CONTENT*(205, HttpStatus.Series.*SUCCESSFUL*, "Reset Content"),  
 *PARTIAL\_CONTENT*(206, HttpStatus.Series.*SUCCESSFUL*, "Partial Content"),  
 *MULTI\_STATUS*(207, HttpStatus.Series.*SUCCESSFUL*, "Multi-Status"),  
 *ALREADY\_REPORTED*(208, HttpStatus.Series.*SUCCESSFUL*, "Already Reported"),  
 *IM\_USED*(226, HttpStatus.Series.*SUCCESSFUL*, "IM Used"),  
 *MULTIPLE\_CHOICES*(300, HttpStatus.Series.*REDIRECTION*, "Multiple Choices"),  
 *MOVED\_PERMANENTLY*(301, HttpStatus.Series.*REDIRECTION*, "Moved Permanently"),  
 *FOUND*(302, HttpStatus.Series.*REDIRECTION*, "Found"),  
 */\*\** ***@deprecated*** *\*/* @Deprecated  
 *MOVED\_TEMPORARILY*(302, HttpStatus.Series.*REDIRECTION*, "Moved Temporarily"),  
 *SEE\_OTHER*(303, HttpStatus.Series.*REDIRECTION*, "See Other"),  
 *NOT\_MODIFIED*(304, HttpStatus.Series.*REDIRECTION*, "Not Modified"),  
 */\*\** ***@deprecated*** *\*/* @Deprecated  
 *USE\_PROXY*(305, HttpStatus.Series.*REDIRECTION*, "Use Proxy"),  
 *TEMPORARY\_REDIRECT*(307, HttpStatus.Series.*REDIRECTION*, "Temporary Redirect"),  
 *PERMANENT\_REDIRECT*(308, HttpStatus.Series.*REDIRECTION*, "Permanent Redirect"),  
 *BAD\_REQUEST*(400, HttpStatus.Series.*CLIENT\_ERROR*, "Bad Request"),  
 *UNAUTHORIZED*(401, HttpStatus.Series.*CLIENT\_ERROR*, "Unauthorized"),  
 *PAYMENT\_REQUIRED*(402, HttpStatus.Series.*CLIENT\_ERROR*, "Payment Required"),  
 *FORBIDDEN*(403, HttpStatus.Series.*CLIENT\_ERROR*, "Forbidden"),  
 *NOT\_FOUND*(404, HttpStatus.Series.*CLIENT\_ERROR*, "Not Found"),  
 *METHOD\_NOT\_ALLOWED*(405, HttpStatus.Series.*CLIENT\_ERROR*, "Method Not Allowed"),  
 *NOT\_ACCEPTABLE*(406, HttpStatus.Series.*CLIENT\_ERROR*, "Not Acceptable"),  
 *PROXY\_AUTHENTICATION\_REQUIRED*(407, HttpStatus.Series.*CLIENT\_ERROR*, "Proxy Authentication Required"),  
 *REQUEST\_TIMEOUT*(408, HttpStatus.Series.*CLIENT\_ERROR*, "Request Timeout"),  
 *CONFLICT*(409, HttpStatus.Series.*CLIENT\_ERROR*, "Conflict"),  
 *GONE*(410, HttpStatus.Series.*CLIENT\_ERROR*, "Gone"),  
 *LENGTH\_REQUIRED*(411, HttpStatus.Series.*CLIENT\_ERROR*, "Length Required"),  
 *PRECONDITION\_FAILED*(412, HttpStatus.Series.*CLIENT\_ERROR*, "Precondition Failed"),  
 *PAYLOAD\_TOO\_LARGE*(413, HttpStatus.Series.*CLIENT\_ERROR*, "Payload Too Large"),  
 */\*\** ***@deprecated*** *\*/* @Deprecated  
 *REQUEST\_ENTITY\_TOO\_LARGE*(413, HttpStatus.Series.*CLIENT\_ERROR*, "Request Entity Too Large"),  
 *URI\_TOO\_LONG*(414, HttpStatus.Series.*CLIENT\_ERROR*, "URI Too Long"),  
 */\*\** ***@deprecated*** *\*/* @Deprecated  
 *REQUEST\_URI\_TOO\_LONG*(414, HttpStatus.Series.*CLIENT\_ERROR*, "Request-URI Too Long"),  
 *UNSUPPORTED\_MEDIA\_TYPE*(415, HttpStatus.Series.*CLIENT\_ERROR*, "Unsupported Media Type"),  
 *REQUESTED\_RANGE\_NOT\_SATISFIABLE*(416, HttpStatus.Series.*CLIENT\_ERROR*, "Requested range not satisfiable"),  
 *EXPECTATION\_FAILED*(417, HttpStatus.Series.*CLIENT\_ERROR*, "Expectation Failed"),  
 *I\_AM\_A\_TEAPOT*(418, HttpStatus.Series.*CLIENT\_ERROR*, "I'm a teapot"),  
 */\*\** ***@deprecated*** *\*/* @Deprecated  
 *INSUFFICIENT\_SPACE\_ON\_RESOURCE*(419, HttpStatus.Series.*CLIENT\_ERROR*, "Insufficient Space On Resource"),  
 */\*\** ***@deprecated*** *\*/* @Deprecated  
 *METHOD\_FAILURE*(420, HttpStatus.Series.*CLIENT\_ERROR*, "Method Failure"),  
 */\*\** ***@deprecated*** *\*/* @Deprecated  
 *DESTINATION\_LOCKED*(421, HttpStatus.Series.*CLIENT\_ERROR*, "Destination Locked"),  
 *UNPROCESSABLE\_ENTITY*(422, HttpStatus.Series.*CLIENT\_ERROR*, "Unprocessable Entity"),  
 *LOCKED*(423, HttpStatus.Series.*CLIENT\_ERROR*, "Locked"),  
 *FAILED\_DEPENDENCY*(424, HttpStatus.Series.*CLIENT\_ERROR*, "Failed Dependency"),  
 *TOO\_EARLY*(425, HttpStatus.Series.*CLIENT\_ERROR*, "Too Early"),  
 *UPGRADE\_REQUIRED*(426, HttpStatus.Series.*CLIENT\_ERROR*, "Upgrade Required"),  
 *PRECONDITION\_REQUIRED*(428, HttpStatus.Series.*CLIENT\_ERROR*, "Precondition Required"),  
 *TOO\_MANY\_REQUESTS*(429, HttpStatus.Series.*CLIENT\_ERROR*, "Too Many Requests"),  
 *REQUEST\_HEADER\_FIELDS\_TOO\_LARGE*(431, HttpStatus.Series.*CLIENT\_ERROR*, "Request Header Fields Too Large"),  
 *UNAVAILABLE\_FOR\_LEGAL\_REASONS*(451, HttpStatus.Series.*CLIENT\_ERROR*, "Unavailable For Legal Reasons"),  
 *INTERNAL\_SERVER\_ERROR*(500, HttpStatus.Series.*SERVER\_ERROR*, "Internal Server Error"),  
 *NOT\_IMPLEMENTED*(501, HttpStatus.Series.*SERVER\_ERROR*, "Not Implemented"),  
 *BAD\_GATEWAY*(502, HttpStatus.Series.*SERVER\_ERROR*, "Bad Gateway"),  
 *SERVICE\_UNAVAILABLE*(503, HttpStatus.Series.*SERVER\_ERROR*, "Service Unavailable"),  
 *GATEWAY\_TIMEOUT*(504, HttpStatus.Series.*SERVER\_ERROR*, "Gateway Timeout"),  
 *HTTP\_VERSION\_NOT\_SUPPORTED*(505, HttpStatus.Series.*SERVER\_ERROR*, "HTTP Version not supported"),  
 *VARIANT\_ALSO\_NEGOTIATES*(506, HttpStatus.Series.*SERVER\_ERROR*, "Variant Also Negotiates"),  
 *INSUFFICIENT\_STORAGE*(507, HttpStatus.Series.*SERVER\_ERROR*, "Insufficient Storage"),  
 *LOOP\_DETECTED*(508, HttpStatus.Series.*SERVER\_ERROR*, "Loop Detected"),  
 *BANDWIDTH\_LIMIT\_EXCEEDED*(509, HttpStatus.Series.*SERVER\_ERROR*, "Bandwidth Limit Exceeded"),  
 *NOT\_EXTENDED*(510, HttpStatus.Series.*SERVER\_ERROR*, "Not Extended"),  
 *NETWORK\_AUTHENTICATION\_REQUIRED*(511, HttpStatus.Series.*SERVER\_ERROR*, "Network Authentication Required");  
  
 private static final HttpStatus[] *VALUES* = *values*();  
 private final int value;  
 private final HttpStatus.Series series;  
 private final String reasonPhrase;  
  
 private HttpStatus(int value, HttpStatus.Series series, String reasonPhrase) {  
 this.value = value;  
 this.series = series;  
 this.reasonPhrase = reasonPhrase;  
 }  
  
 public int value() {  
 return this.value;  
 }  
  
 public HttpStatus.Series series() {  
 return this.series;  
 }  
  
 public String getReasonPhrase() {  
 return this.reasonPhrase;  
 }  
  
 public boolean is1xxInformational() {  
 return this.series() == HttpStatus.Series.*INFORMATIONAL*;  
 }  
  
 public boolean is2xxSuccessful() {  
 return this.series() == HttpStatus.Series.*SUCCESSFUL*;  
 }  
  
 public boolean is3xxRedirection() {  
 return this.series() == HttpStatus.Series.*REDIRECTION*;  
 }  
  
 public boolean is4xxClientError() {  
 return this.series() == HttpStatus.Series.*CLIENT\_ERROR*;  
 }  
  
 public boolean is5xxServerError() {  
 return this.series() == HttpStatus.Series.*SERVER\_ERROR*;  
 }  
  
 public boolean isError() {  
 return this.is4xxClientError() || this.is5xxServerError();  
 }  
  
 public String toString() {  
 return this.value + " " + this.name();  
 }  
  
 public static HttpStatus valueOf(int statusCode) {  
 HttpStatus status = *resolve*(statusCode);  
 if (status == null) {  
 throw new IllegalArgumentException("No matching constant for [" + statusCode + "]");  
 } else {  
 return status;  
 }  
 }  
  
 @Nullable  
 public static HttpStatus resolve(int statusCode) {  
 HttpStatus[] var1 = *VALUES*;  
 int var2 = var1.length;  
  
 for(int var3 = 0; var3 < var2; ++var3) {  
 HttpStatus status = var1[var3];  
 if (status.value == statusCode) {  
 return status;  
 }  
 }  
  
 return null;  
 }  
  
 public static enum Series {  
 *INFORMATIONAL*(1),  
 *SUCCESSFUL*(2),  
 *REDIRECTION*(3),  
 *CLIENT\_ERROR*(4),  
 *SERVER\_ERROR*(5);  
  
 private final int value;  
  
 private Series(int value) {  
 this.value = value;  
 }  
  
 public int value() {  
 return this.value;  
 }  
  
 */\*\** ***@deprecated*** *\*/* @Deprecated  
 public static HttpStatus.Series valueOf(HttpStatus status) {  
 return status.series;  
 }  
  
 public static HttpStatus.Series valueOf(int statusCode) {  
 HttpStatus.Series series = *resolve*(statusCode);  
 if (series == null) {  
 throw new IllegalArgumentException("No matching constant for [" + statusCode + "]");  
 } else {  
 return series;  
 }  
 }  
  
 @Nullable  
 public static HttpStatus.Series resolve(int statusCode) {  
 int seriesCode = statusCode / 100;  
 HttpStatus.Series[] var2 = *values*();  
 int var3 = var2.length;  
  
 for(int var4 = 0; var4 < var3; ++var4) {  
 HttpStatus.Series series = var2[var4];  
 if (series.value == seriesCode) {  
 return series;  
 }  
 }  
  
 return null;  
 }  
 }  
}