```
/*-------------------------------------------------------------------------------------------------
 * Author:      Dan Cassidy
 * Date:        2015-08-04
 * Assignment:  Project
 * Source File: MachineType.java
 * Language:    Java
 * Course:      CSCI-C 490, Android Programming, MoWe 08:00
-------------------------------------------------------------------------------------------------*/
package com.chaoticcognitions.aenigma.models.machines;

import com.chaoticcognitions.aenigma.models.rotors.RotorType;

/**
 * Enum to store the relevant information about the different types of Enigma machines in a single
 * place.
 * @author Dan Cassidy
 */
public enum MachineType {
    ENIGMA_I,
    NORWAY_ENIGMA,
    ENIGMA_M3,
    ENIGMA_M4,
    ENIGMA_G,
    ENIGMA_D,
    ENIGMA_K,
    SWISS_K,
    ENIGMA_KD,
    RAILWAY_ENIGMA,
    ENIGMA_T;

    /**
     * Gets a list of possible stators for the given machine type.
     * @return The possible stators for the given machine type.
     */
    public RotorType[] possibleStators() {
        switch (this) {
            case ENIGMA_I:
                return new RotorType[]{RotorType.I_ETW};
            case NORWAY_ENIGMA:
                return new RotorType[]{RotorType.N_ETW};
            case ENIGMA_M3:
                return new RotorType[]{RotorType.M3_ETW};
            case ENIGMA_M4:
                return new RotorType[]{RotorType.M4_ETW};
            case ENIGMA_G:
                return new RotorType[]{RotorType.G_ETW};
            case ENIGMA_D:
                return new RotorType[]{RotorType.D_ETW};
            case ENIGMA_K:
                return new RotorType[]{RotorType.K_ETW};
            case SWISS_K:
                return new RotorType[]{RotorType.KS_ETW};
            case ENIGMA_KD:
                return new RotorType[]{RotorType.KD_ETW};
            case RAILWAY_ENIGMA:
                return new RotorType[]{RotorType.R_ETW};
            case ENIGMA_T:
                return new RotorType[]{RotorType.T_ETW};

            default:
```

```
61                  return new RotorType[]{};
62              }
63          }
64
65          /**
66           * Gets a list of possible rotors for the given machine type.
67           * @return The possible rotors for the given machine type.
68           */
69          public RotorType[] possibleRotors() {
70              switch (this) {
71                  case ENIGMA_I:
72                      return new RotorType[]{RotorType.I_I, RotorType.I_II, RotorType.I_III,
                        RotorType.I_IV, RotorType.I_V};
73                  case NORWAY_ENIGMA:
74                      return new RotorType[]{RotorType.N_I, RotorType.N_II, RotorType.N_III,
                        RotorType.N_IV, RotorType.N_V};
75                  case ENIGMA_M3:
76                      return new RotorType[]{RotorType.M3_I, RotorType.M3_II, RotorType.M3_III,
                        RotorType.M3_IV, RotorType.M3_V, RotorType.M3_VI, RotorType.M3_VII,
                        RotorType.M3_VIII};
77                  case ENIGMA_M4:
78                      return new RotorType[]{RotorType.M4_I, RotorType.M4_II, RotorType.M4_III,
                        RotorType.M4_IV, RotorType.M4_V, RotorType.M4_VI, RotorType.M4_VII,
                        RotorType.M4_VIII, RotorType.M4_BETA, RotorType.M4_GAMMA};
79                  case ENIGMA_G:
80                      return new RotorType[]{RotorType.G_I, RotorType.G_II, RotorType.G_III};
81                  case ENIGMA_D:
82                      return new RotorType[]{RotorType.D_I, RotorType.D_II, RotorType.D_III};
83                  case ENIGMA_K:
84                      return new RotorType[]{RotorType.K_I, RotorType.K_II, RotorType.K_III};
85                  case SWISS_K:
86                      return new RotorType[]{RotorType.KS_I, RotorType.KS_II, RotorType.KS_III};
87                  case ENIGMA_KD:
88                      return new RotorType[]{RotorType.KD_I, RotorType.KD_II, RotorType.KD_III};
89                  case RAILWAY_ENIGMA:
90                      return new RotorType[]{RotorType.R_I, RotorType.R_II, RotorType.R_III};
91                  case ENIGMA_T:
92                      return new RotorType[]{RotorType.T_I, RotorType.T_II, RotorType.T_III,
                        RotorType.T_IV, RotorType.T_V, RotorType.T_VI, RotorType.T_VII, RotorType.T_VIII};
93
94                  default:
95                      return new RotorType[]{};
96              }
97          }
98
99          /**
100          * Gets a list of possible reflectors for the given machine type.
101          * @return The possible reflectors for the given machine type.
102          */
103         public RotorType[] possibleReflectors(){
104             switch (this) {
105                 case ENIGMA_I:
106                     return new RotorType[]{RotorType.I_UKW_A, RotorType.I_UKW_B, RotorType.I_UKW_C};
107                 case NORWAY_ENIGMA:
108                     return new RotorType[]{RotorType.N_UKW};
109                 case ENIGMA_M3:
110                     return new RotorType[]{RotorType.M3_UKW_B, RotorType.M3_UKW_C};
111                 case ENIGMA_M4:
112                     return new RotorType[]{RotorType.M4_UKW_B, RotorType.M4_UKW_C};
113                 case ENIGMA_G:
```

```
114                 return new RotorType[]{RotorType.G_UKW};
115             case ENIGMA_D:
116                 return new RotorType[]{RotorType.D_UKW};
117             case ENIGMA_K:
118                 return new RotorType[]{RotorType.K_UKW};
119             case SWISS_K:
120                 return new RotorType[]{RotorType.KS_UKW};
121             case ENIGMA_KD:
122                 return new RotorType[]{RotorType.KD_UKW};
123             case RAILWAY_ENIGMA:
124                 return new RotorType[]{RotorType.R_UKW};
125             case ENIGMA_T:
126                 return new RotorType[]{RotorType.T_UKW};
127
128             default:
129                 return new RotorType[]{};
130         }
131     }
132
133     /**
134      * Gets whether the machine is Enigma stepped or not based on the type.
135      * @return The possible stators for the given machine type.
136      */
137     public boolean isEnigmaStepped() {
138         switch (this) {
139             case ENIGMA_I:
140             case NORWAY_ENIGMA:
141             case ENIGMA_M3:
142             case ENIGMA_M4:
143             case ENIGMA_D:
144             case ENIGMA_K:
145             case SWISS_K:
146             case ENIGMA_KD:
147             case RAILWAY_ENIGMA:
148             case ENIGMA_T:
149                 return true;
150             case ENIGMA_G:
151                 return false;
152
153             default:
154                 return true;
155         }
156     }
157
158     /**
159      * Gets whether the machine's reflector is visible or not based on the type.
160      * @return Whether the machine's reflector is visible.
161      */
162     public boolean hasVisibleReflector() {
163         switch (this) {
164             case ENIGMA_I:
165             case NORWAY_ENIGMA:
166             case ENIGMA_M3:
167             case ENIGMA_M4:
168                 return false;
169             case ENIGMA_G:
170             case ENIGMA_D:
171             case ENIGMA_K:
172             case SWISS_K:
173             case ENIGMA_KD:
```

```
174             case RAILWAY_ENIGMA:
175             case ENIGMA_T:
176                 return true;
177 
178             default:
179                 return false;
180         }
181     }
182 
183     //TODO create method comment
184     public boolean hasPlugboard() {
185         switch (this) {
186             case ENIGMA_I:
187             case NORWAY_ENIGMA:
188             case ENIGMA_M3:
189             case ENIGMA_M4:
190                 return true;
191             case ENIGMA_G:
192             case ENIGMA_D:
193             case ENIGMA_K:
194             case SWISS_K:
195             case ENIGMA_KD:
196             case RAILWAY_ENIGMA:
197             case ENIGMA_T:
198                 return false;
199 
200             default:
201                 return false;
202         }
203     }
204 
205     /**
206      * Gets the number of rotors a machine has based on its type. Note that this is only describing
207      * the number of actual rotors and does not include the stator or reflector.
208      * @return The number of rotors the machine has.
209      */
210     public int numberOfRotors() {
211         switch (this) {
212             case ENIGMA_I:
213             case NORWAY_ENIGMA:
214             case ENIGMA_M3:
215             case ENIGMA_G:
216             case ENIGMA_D:
217             case ENIGMA_K:
218             case SWISS_K:
219             case ENIGMA_KD:
220             case RAILWAY_ENIGMA:
221             case ENIGMA_T:
222                 return 3;
223             case ENIGMA_M4:
224                 return 4;
225 
226             default:
227                 return 3;
228         }
229     }
230 
231     /**
232      * Returns the string representation of the machine type.
233      * @return The string representation of the machine type.
```

```
234         */
235        @Override public String toString() {
236            switch (this) {
237                case ENIGMA_I:
238                    return "Enigma I";
239                case NORWAY_ENIGMA:
240                    return "Norway Enigma";
241                case ENIGMA_M3:
242                    return "Enigma M3";
243                case ENIGMA_M4:
244                    return "Enigma M4";
245                case ENIGMA_G:
246                    return "Enigma G";
247                case ENIGMA_D:
248                    return "Enigma D";
249                case ENIGMA_K:
250                    return "Enigma K";
251                case SWISS_K:
252                    return "Swiss-K";
253                case ENIGMA_KD:
254                    return "Enigma KD";
255                case RAILWAY_ENIGMA:
256                    return "Railway Enigma";
257                case ENIGMA_T:
258                    return "Enigma T";
259
260                default:
261                    return "Unknown";
262            }
263        }
264    }
265
```