```
1. public class Assistant {
2.
3.     private String email;
4.
5.     private String name;
6.
7.     private static Assistant[] assistants = new
   Assistant[100];
8.
9.     private static int numAssistants = 0;
10.
11.         /**
12.          * The constructor for the Assistant class.
13.          * This is a class used for people related to the
   university (staff or students)
14.          * who are volunteering to perform COVID tests.
15.          *
16.          * @param email the university email of the
   assistant.
17.          * @param name the name of the assistant.
18.          */
19.         public Assistant(String email, String name){
20.             this.email = checkEmail(email);
21.             this.name = checkName(name);
22.             addAssistants(this);
23.             iterateNumAssistants();
24.         }
25.
26.         /**
27.          * The email getter method.
28.          *
29.          * @return the university email of a specific
   assistant.
30.          */
31.         public String getEmail(){
32.             return email;
33.         }
34.
35.         /**
36.          * The name getter method
37.          *
38.          * @return the name of a specific assistant.
39.          */
40.         public String getName(){
41.             return name;
42.         }
43.
44.         /**
45.          * The array of the all assistants static getter
   method.
46.          *
47.          * @return the array of all assistants.
48.          */
49.         public static Assistant[] getAssistants(){
```

```
50.             return assistants;
51.         }
52.
53.         /**
54.          * The number of assistants static getter method.
55.          *
56.          * @return the number of assistants.
57.          */
58.         public static int getnumAssistants(){
59.             return numAssistants;
60.         }
61.
62.         /**
63.          * The email setter method.
64.          *
65.          * @param email the university email of a specific
   assistant.
66.          */
67.         public void setEmail(String email){
68.             this.email = checkEmail(email);
69.         }
70.
71.         /**
72.          * The name setter method.
73.          *
74.          * @param name the name of a specific assistant.
75.          */
76.         public void setName(String name){
77.             this.name = checkName(name);
78.         }
79.
80.         /**
81.          * The number of assistants iterator private
   static method.
82.          * Increases the number of assistants by 1 when
   called.
83.          */
84.         private static void iterateNumAssistants(){
85.             numAssistants += 1;
86.         }
87.
88.         /**
89.          * The add assistants to the array of assistants
   static method
90.          *
91.          * @param assistant an instance of an assistant.
92.          */
93.         private static void addAssistants(Assistant
   assistant){
94.             assistants[numAssistants] = assistant;
95.         }
96.
97.         /**
98.          * The name checker private method.
```

```
99.          * Checks if the name of an assistant is not empty
   or blank spaces.
100.         *
101.         * @param name the name of a specific assistant.
102.         * @return the name of a specific assistant.
103.         */
104.        private String checkName(String name){
105.            if (name.trim().isEmpty()){
106.                throw new IllegalArgumentException("The
   name string should have at least one character that is not a
   space.");
107.            }
108.            return name;
109.        }
110.
111.        /**
112.         * The email checker private method.
113.         * Checks if the email of an assistant ends with
   "@uok.ac.uk" and is unique.
114.         *
115.         * @param email the email of a specific assistant.
116.         * @return the email of a specific assistant.
117.         */
118.        private String checkEmail(String email){
119.            if (email.endsWith("@uok.ac.uk")){
120.                for (int i=0;i<numAssistants;i++){
121.                    if
   (assistants[i].getEmail().equals(email)){
122.                        throw new
   IllegalArgumentException("The email should be unique.");
123.                    }
124.                }
125.                return email;
126.            }
127.            throw new IllegalArgumentException("The email
   string should always end with @uok.ac.uk.");
128.        }
129.
130.        /**
131.         * A method to return all the assistants as a
   string.
132.         *
133.         * @return a string of all the assistants.
134.         */
135.        public static String toStringAll(){
136.            String allAssistants = "Assistants-\n";
137.            for (int i=0;i<numAssistants;i++){
138.                allAssistants =
   allAssistants.concat((i+11)+". "+assistants[i].toString()
   +"\n");
139.            }
140.            return allAssistants;
141.        }
142.
```

```
143.         /**
144.          * A method to return the string of an assistant
145.          *
146.          * @return a string of an assistant.
147.          */
148.         public String toString(){
149.             return "| "+name+" | "+email+" |";
150.         }
151.     }
152.
153.
```