

[Practice](#)[Compete](#)[Jobs](#)[Rank](#)[Leaderboard](#)[Dashboard](#) > [Java](#) > [Advanced](#) > [Java Factory Pattern](#)

Points: 728 Rank: 2692

Java Factory Pattern

 by [Shafaet](#)

Problem

Submissions

Leaderboard

Discussions

Editorial

According to Wikipedia, a factory is simply an object that returns another object from some other method call, which is assumed to be "new".

In this problem, you are given an interface *Food*. There are two classes *Pizza* and *Cake* which implement the *Food* interface, and they both contain a method *getType()*.

The main function in the *Main* class creates an instance of the *FoodFactory* class. The *FoodFactory* class contains a method *getFood(String)* that returns a new instance of *Pizza* or *Cake* according to its parameter.

You are given the partially completed code in the editor. Please complete the *FoodFactory* class.

Sample Input 1

cake

Sample Output 1

The factory returned class Cake
Someone ordered a Dessert!

Sample Input 2

pizza

Sample Output 2

The factory returned class Pizza
Someone ordered Fast Food![f](#) [t](#) [in](#)Submissions: [11889](#)

Max Score: 15

Difficulty: Easy

Rate This Challenge:

☆☆☆☆☆

[More](#)

Current Buffer (saved locally, editable)

Java 7



```
1 ▶ import ↔;  
3 ▼ interface Food {
```

```

4     public String getType();
5     }
6     class Pizza implements Food {
7     public String getType() {
8         return "Someone ordered a Fast Food!";
9     }
10    }
11
12    class Cake implements Food {
13
14    public String getType() {
15        return "Someone ordered a Dessert!";
16    }
17    }
18    class FoodFactory {
19        public Food getFood(String order) {
20
21

```

```

22    \Write your code here
23

```

```

24    }//End of getFood method
25
26    }//End of factory class
27
28    public class Solution {
29
30    public static void main(String args[]){
31        Do_Not_Terminate.forbidExit();
32
33    try{
34
35        Scanner sc=new Scanner(System.in);
36        //creating the factory
37        FoodFactory foodFactory = new FoodFactory();
38
39        //factory instantiates an object
40        Food food = foodFactory.getFood(sc.nextLine());
41
42
43        System.out.println("The factory returned "+food.getClass());
44        System.out.println(food.getType());
45    }
46    catch (Do_Not_Terminate.ExitTrappedException e) {
47        System.out.println("Unsuccessful Termination!!");
48    }
49    }
50
51    }
52    class Do_Not_Terminate {
53
54    public static class ExitTrappedException extends SecurityException {
55
56        private static final long serialVersionUID = 1L;
57    }
58
59    public static void forbidExit() {
60        final SecurityManager securityManager = new SecurityManager() {
61            @Override
62            public void checkPermission(Permission permission) {
63                if (permission.getName().contains("exitVM")) {

```

```
64  throw new ExitTrappedException();
65  }
66  }
67  };
68  System.setSecurityManager(securityManager);
69  }
70  }
71
72
73
74
75
76
```

Line: 4 Col: 1

 [Upload Code as File](#)

☐ Test against custom input

Run Code

Submit Code

Join us on IRC at [#hackerrank](#) on freenode for hugs or bugs.

[Contest Calendar](#) | [Blog](#) | [Scoring](#) | [Environment](#) | [FAQ](#) | [About Us](#) | [Support](#) | [Careers](#) | [Terms Of Service](#) | [Privacy Policy](#) | [Request a Feature](#)