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!
```

```
     public String getType();
    }
    class Pizza implements Food {
     public String getType() {
     return "Someone ordered a Fast Food!";
     }
    }

    class Cake implements Food {

     public String getType() {
     return "Someone ordered a Dessert!";
     }
    }
    class FoodFactory {
        public Food getFood(String order) {


\Write your code here



}//End of getFood method

    }//End of factory class

    public class Solution {

     public static void main(String args[]){
            Do_Not_Terminate.forbidExit();

        try{

            Scanner sc=new Scanner(System.in);
            //creating the factory
            FoodFactory foodFactory = new FoodFactory();

            //factory instantiates an object
            Food food = foodFactory.getFood(sc.nextLine());


            System.out.println("The factory returned "+food.getClass());
            System.out.println(food.getType());
        }
        catch (Do_Not_Terminate.ExitTrappedException e) {
            System.out.println("Unsuccessful Termination!!");
        }
     }

    }
    class Do_Not_Terminate {

        public static class ExitTrappedException extends SecurityException {

            private static final long serialVersionUID = 1L;
        }

        public static void forbidExit() {
            final SecurityManager securityManager = new SecurityManager() {
                @Override
                public void checkPermission(Permission permission) {
                    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