第一种实现：

```
public class Multiton1 {
    private static int maxCount = 1;
    private Multiton1() { System.out.print("New instance!\n"); }
    private static class HolderClass {
        private static List instanceList = new ArrayList();
        static {
            for (int i = 0; i < maxCount; i++) {
                Multiton1 instance = new Multiton1();
                instanceList.add(instance);
            }
        }
    }
    public static void setMaxCount(int max) {
        maxCount = max;
    }
    public static Multiton1 getInstance() {
        int i = new Random().nextInt(maxCount);
        Multiton1 instance = (Multiton1)HolderClass.instanceList.get(i);
        System.out.println("Got Instance No." + (i + 1));
        return instance;
    }
}
```

思路：多例类 default 实例个数为 1，如果用户自行指定，则必须在第一次调用 getInstance()之前直接对 Multiton 类调用 setMaxCount(max)指定，否则一旦调用过 getInstance()就不能再次更改。具体实现模仿静态内部类方法，在首次调用 getInstance()时，通过内部类 HolderClass 一次性创建 maxCount 个实例储存在列表中，每次程序调用 getInstance()，便随机返回其中一个。

```
public class Client1 {
    public static void main(String args[]) {
        Multiton1.setMaxCount(10);
        Multiton1 M1 = Multiton1.getInstance();
        Multiton1 M2 = Multiton1.getInstance();
        Multiton1 M3 = Multiton1.getInstance();
        Multiton1 M4 = Multiton1.getInstance();
        Multiton1 M5 = Multiton1.getInstance();
    }
}
```

测试：让默认构造函数 Multiton()每次被调用时输出一句 New Instance!，获取实例的函数 getInstance()每次输出 Got Instance + 获取第几个实例的编号，注意到这里一次性输出了 10 次 New Instance!，然后是随机获取的五个实例。

```
New instance!
New instance!
New instance!
New instance!
New instance!
New instance!
New instance!
New instance!
New instance!
New instance!
Got Instance No.9
Got Instance No.8
Got Instance No.8
Got Instance No.10
Got Instance No.5
```

第二种实现：

```
public class Multiton2 {
    private static int maxCount = 1;
    private static int currentCount = 0;
    private static List instanceList = new ArrayList();
    private Multiton2() { System.out.print("New instance!\n"); }
    public static void setMaxCount(int max) {
        maxCount = max;
    }
    synchronized public static Multiton2 getInstance() {
        if (currentCount < maxCount) {
            Multiton2 instance = new Multiton2();
            instanceList.add(instance);
            System.out.println("Got Instance No." + (++currentCount));
            return instance;
        }
        else {
            int i = new Random().nextInt(maxCount);
            Multiton2 instance = (Multiton2)instanceList.get(i);
            System.out.println("Got Instance No." + (i + 1));
            return instance;
        }
    }
}
```

思路：实例个数指定方法同上。具体实现模仿懒汉式单例类与锁方法，每次调用 getInstance()时，检查当前已经创建的实例个数 currentCount 是否小于最大允许创建的实例个数 maxCount，若是，创建新的实例存入 instanceList 并返回，同时更新 currentCount；若不是，随机返回 List 中的一个实例。

```
public class Client2 {
    public static void main(String args[]) {
        Multiton2.setMaxCount(3);
        Multiton2 M1 = Multiton2.getInstance();
        Multiton2 M2 = Multiton2.getInstance();
        Multiton2 M3 = Multiton2.getInstance();
        Multiton2 M4 = Multiton2.getInstance();
        Multiton2 M5 = Multiton2.getInstance();
    }
}
```

测试：原理同上，前三次调用 getInstance()每次输出一句 New Instance!然后返回刚刚创建的实例（No.1, No.2, No.3），后两次没有 New Instance!而且返回的是之前三次创建的实例，可见两种实现方法的效果都符合设想。

```
New instance!
Got Instance No.1
New instance!
Got Instance No.2
New instance!
Got Instance No.3
Got Instance No.2
Got Instance No.1
```