

Singleton

Scenario:

A company wants to build a **PrinterManager** class to manage a number of printers available in the company. This **PrinterManager** is the only instance and entry point for anyone who wants to print a document with any printers in the company. After a print job is sent to the **PrinterManager**, the **PrinterManager** will check whether a connected printer is available. If there is an available printer, the **PrinterManager** will send the job to it. If all the printers are busy, an error message is returned. Since the system is still in prototyping phase, if a printer is assigned a print job, it will stay as unavailable state, unless the **PrinterManager** resets it.

Objective:

Please use Singleton pattern to implement **PrinterManager** class, provide necessary functions based on the scenario described above. Create a client to demonstrate the use of it.

Tasks: (You are encouraged to work with a partner)

1. The **Printer** class is already implemented (code is available in Blackboard). Import it in to your package.
2. Create a new class called **PrinterManager**.
3. In **PrinterManager**, define a private static "single instance", define an array of **Printer** called **printers** to store available printers.

```
private static PrinterManager singleInstance;  
private Printer[] printers;  
private final int SIZE = 8;
```

4. Create a private constructor for **PrinterManager**.

// private constructor

```
private PrinterManager() {  
    printers = new Printer[SIZE];  
    for (int i = 0; i < SIZE; i++) {  
        printers[i] = new Printer();  
    }  
}
```

5. Create a public static accessor `getInstance()` method in **PrinterManager**.

// important method of returning singleton instance

```
public static PrinterManager getInstance() {  
    if (singleInstance == null) {  
        singleInstance = new PrinterManager();  
    }  
    return singleInstance;  
}
```

6. Create a method called **assignJob**. It is able to assign a job to an available printer.

// assign a job to printer

```
public void assignJob(String printJob) {  
    boolean allPrintersBusy = true;  
    for (int i = 0; i < printers.length; i++) {  
        if (!printers[i].isBusy()) {  
            allPrintersBusy = false;  
            printers[i].setPrintJob(printJob);  
            printers[i].setBusy(isBusy:true);  
            break;  
        }  
    }  
  
    if (allPrintersBusy) {  
        System.out.println(x:"All printers were busy!");  
    }  
}
```

Output shown on Question 8

7. Create a method called **showStatus**, which shows the status of every printer.

```
public void showStatus() {  
    for (int i = 0; i < printers.length; i++) {  
        String message = "Printer " + i + " " + (printers[i].isBusy() ? "Status: Busy" : "Status: Available");  
        System.out.println(message);  
    }  
}
```

Output shown on Question 8

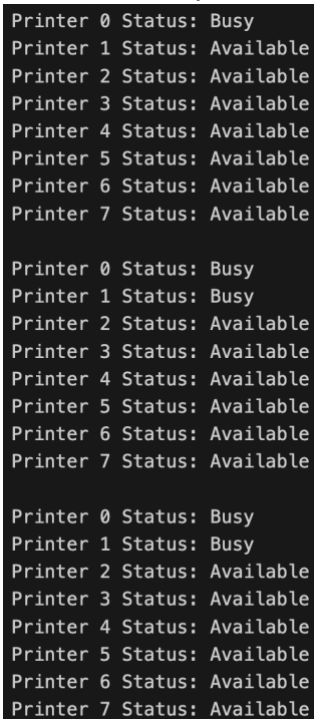
8. Create a **Client** class and use the following **main** method to test your implementation.

```
public static void main(String[] args) {  
    // get the singleton instance  
    PrinterManager printerManager = PrinterManager.getInstance();  
}
```

Name: E5_David_Mugwaneza

```
// assign some job
printerManager.assignJob("print something");
printerManager.assignJob("print something again");
// show status
printerManager.showStatus();

// check if you can get another instance
PrinterManager printerManager2 = PrinterManager.getInstance();
// show status
printerManager2.showStatus();
}
```



```
Printer 0 Status: Busy
Printer 1 Status: Available
Printer 2 Status: Available
Printer 3 Status: Available
Printer 4 Status: Available
Printer 5 Status: Available
Printer 6 Status: Available
Printer 7 Status: Available

Printer 0 Status: Busy
Printer 1 Status: Busy
Printer 2 Status: Available
Printer 3 Status: Available
Printer 4 Status: Available
Printer 5 Status: Available
Printer 6 Status: Available
Printer 7 Status: Available

Printer 0 Status: Busy
Printer 1 Status: Busy
Printer 2 Status: Available
Printer 3 Status: Available
Printer 4 Status: Available
Printer 5 Status: Available
Printer 6 Status: Available
Printer 7 Status: Available
```

9. You are welcome to add additional methods/implementation based on the scenarios.

What to turn in?

1. only one submission is required on Blackboard (please write both names)
2. a pdf file with title as “E5_lastname1_lastname2.pdf”, lastname1 and lastname2 are the last name of the two students who finished the assignment together. The pdf file should include the Screenshot each function and paste to the corresponding questions and screenshot your executed results as well.
3. a .zip file with title as “E1_lastname1_lastname2.zip”, inside the .zip, you should have all source code of your programming solutions.

Example Submission:

Student1: Alice Brown

Student2: Tom Jackson

.pdf file: E5_Brown_Jackson.pdf

Name: E5_David_Mugwaneza

1. The **Printer** class is already implemented (code is available in Blackboard). Import it in to your package.

Screenshot for this function

Similar for all other questions 2, 3, ...

Finally:

Screenshot for output

.zip file: E5_Brown_Jackson.zip

Source code for programming task 1 and 2.