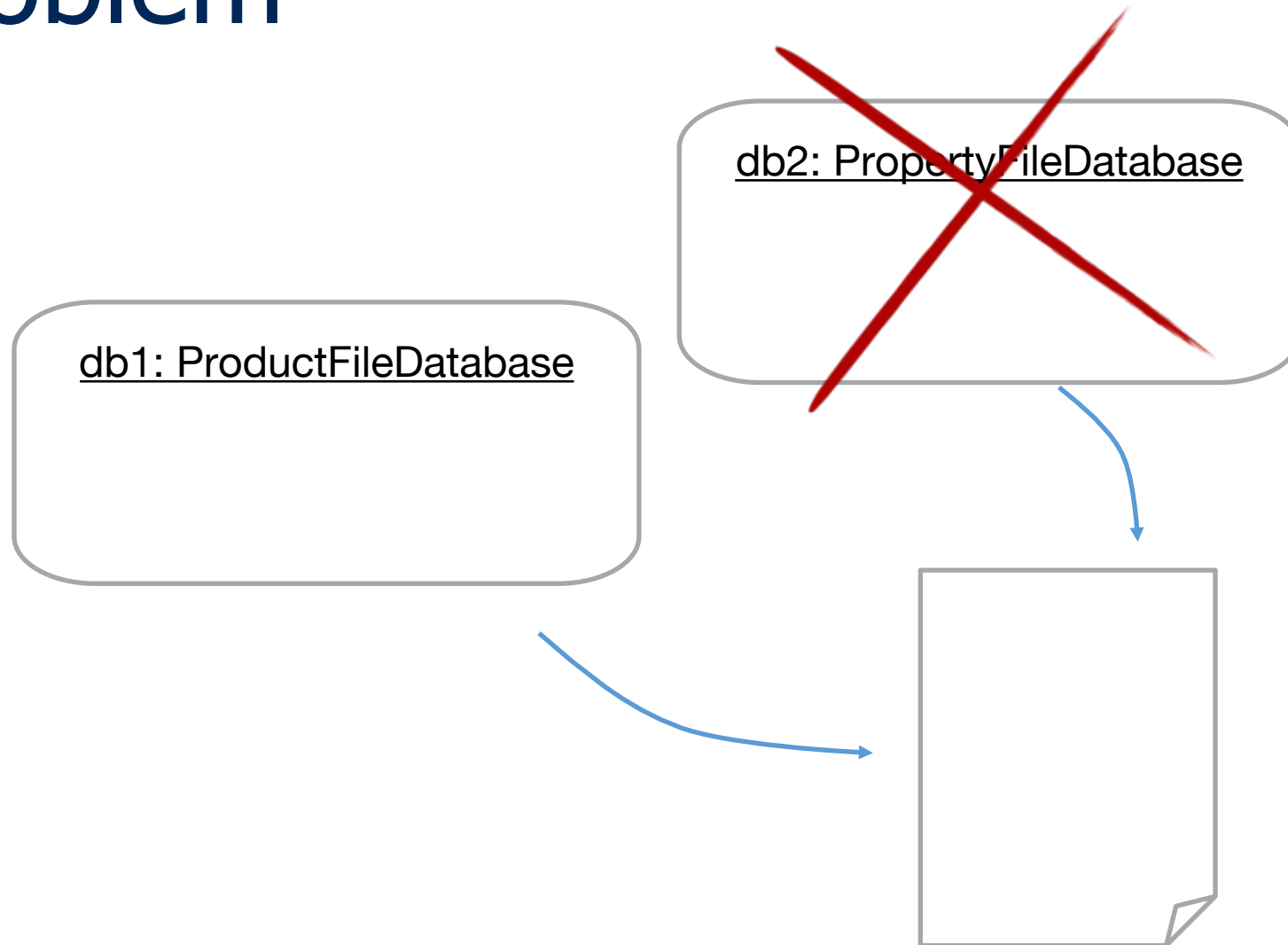


Problem



Solution?

- Disable object creation
→ private constructor
- We still need one object...





UC Leuven
Limburg

MOVING MINDS

Singleton



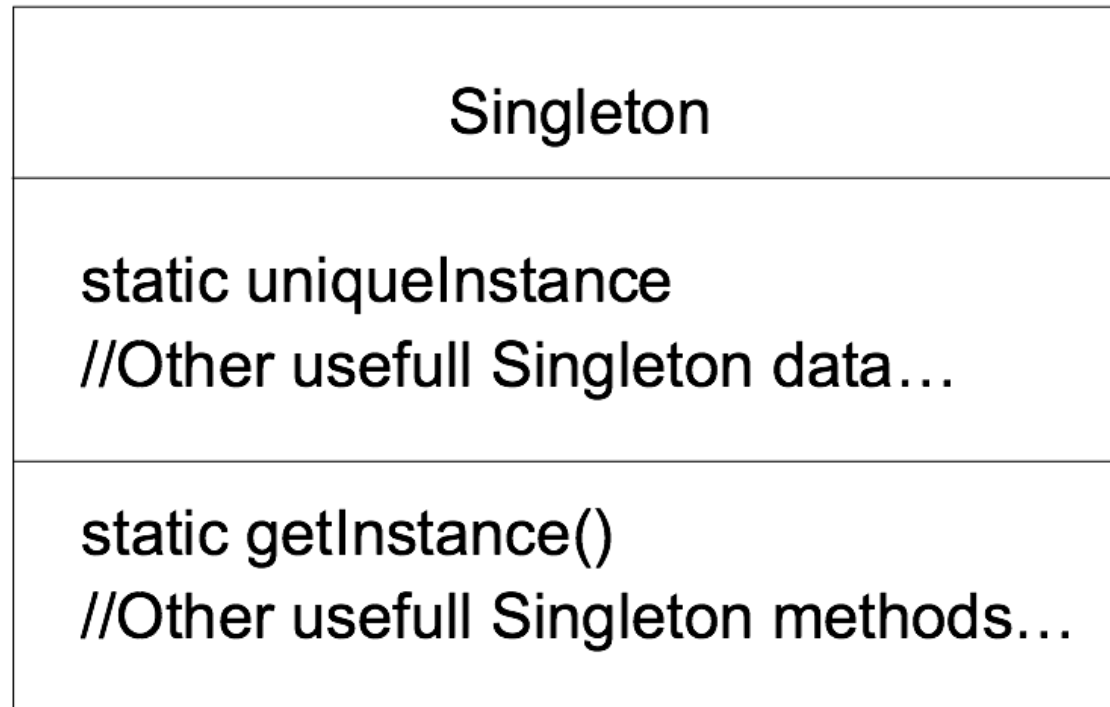
Why?

- Only ONE instance of a class
- that is globally accessible

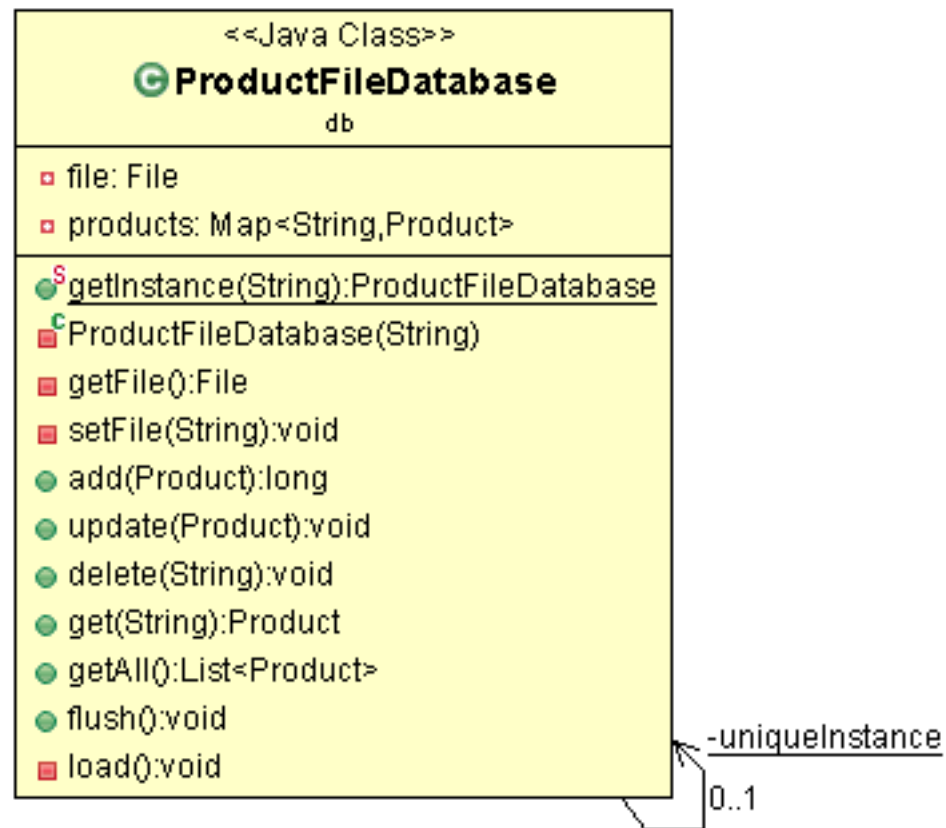
How?

- Static method
- that calls private constructor
- making sure only one instance is created

Class diagram



ProductFileDatabase



Example


```
public class Singleton {  
    private static Singleton uniqueInstance;  
    // other useful instance variables  
  
    private Singleton () {}  
  
    public static Singleton getInstance () {  
        if (uniqueInstance == null) {  
            uniqueInstance = new Singleton ();  
        }  
        return uniqueInstance;  
    }  
  
    // other useful methods  
}
```

*What if 2 clients
access this code
at the same time?!*

Multithreading: solution 1

```
public class Singleton {  
    private static Singleton uniqueInstance;  
  
    private Singleton () {}  
  
    public static synchronized Singleton getInstance() {  
        if (uniqueInstance == null) {  
            uniqueInstance = new Singleton ();  
        }  
        return uniqueInstance;  
    }  
}
```

Expensive!



Multithreading: solution 2

```
public class Singleton {
```

```
    private static Singleton uniqueInstance =  
        new Singleton();
```

```
    private Singleton () {}
```

once *Not always needed*

```
    public static Singleton getInstance () {
```

```
        return uniqueInstance;
```

```
    }
```

```
}
```

only return

Multithreading: solution 3

```
public class Singleton {  
    private volatile static Singleton uniqueInstance;  
  
    private Singleton () {}  
    public static Singleton getInstance () {  
        if (uniqueInstance == null) {  
            synchronized (Singleton.class) {  
                if (uniqueInstance == null) {  
                    uniqueInstance = new Singleton ();  
                }  
            }  
        }  
        return uniqueInstance;  
    }  
}
```

once →

↓
only when needed

just
another
example

Singleton vs. design principles



SOLID

- SRP:
 - Not guaranteed
- OCP:
 - NOK → use only when needed
 - private constructor → subclass not possible
- ISP en DIP:
 - Not applicable

?

