COMP2511

23T1 Week 8

WEDNESDAY 1PM - 4PM (W13B)

FRIDAY 11AM - 2PM (F11A)

Slides by Alvin Cherk (z5311001)

This week

- Generic programming
- Singleton pattern

Please let me know in the lab if you want to do assignment-iii as a pair. If not, you are default solo

Generic Programming

Generic Programming

What are generics?

Generics enable types to be passed when defining classes, interfaces or methods

- Remove casting and offer stronger type checks at compile time
- Allow implementations of **generic algorithms**, that work on a collection of **different types**
- Adds stability to code by making more of your bugs detectable at run-time

Generic Programming

```
1  /**
2   * How many different types of T here?
3   *
4   * It's all about the scope
5   */
6  public class ConfusingClass<T> {
7      private T t1;
8
9      public <T> T t2(T t) {
10         return null;
11     }
12
13     public static <T> T t3(T t) {
14         return null;
15     }
16
17     public static void main(String[] args) {
18         ConfusingClass<String> cc = new ConfusingClass<String>();
19         cc.t2(3);
20     }
21 }
```

Example from Webster

Generic Programming

```
1  class ConfusingClass<T1> {
2      private T1 t1;
3
4      // private T2 t2; // nope
5      // private T3 t3; // nope
6
7      public <T2> T2 t2(T2 t) {
8          T1 t1;
9          T2 t2;
10         // T3 t3; // nope
11         return null;
12     }
13
14     // ConfusingClass.t3(...)
15     public static <T3> T3 t3(T3 t) {
16         // T1 t1; // nope
17         // T2 t2; // nope
18         T3 t3;
19         return null;
20     }
21 }
```

Example from Webster

Code Demo

Stack.java

Code Demo

Inside **`src/stack`**, there are a series of stubs for a **`Stack`** class which takes in a generic type. There are a series of tests inside **`StackTest.java`** which currently fail.

Implement the methods so that the tests pass, using an **`ArrayList`** to store the internal data structure. Answer the following questions:

1. What is E?
    - A generic type
2. What is the **`Iterable`** interface? Why does it have an `E` as well? What methods does it force us to implement?
    - Iterable: Something that can be iterated over
    - Forces us to implement the **`.iterator()`** method

Code Demo

1. When completing **`toArrayList`**, why do we need to make a copy rather then just returning our internal **`ArrayList`**?
    - Don't want to break encapsulation
2. What does the **`.iterator()`** method allow us to do? Look at **`StackTest.java`**
    - Allows us to loop through it like a normal collection in a standardized way

Code Demo

```
1 public static Integer sumStack(Stack<? extends Integer> stack);
```

1. What does the **<? extends Type>** and **<? super Type>** mean?
    - **extends** : the parameterized type must be a class or subclass of the given type
    - **super** : the parameterised type must be a class or super class of the given type
2. What is the difference between **?** and **E**?
    - **?** can't be referred to as a type (Type erasure)
    - **E** can

Singleton Pattern

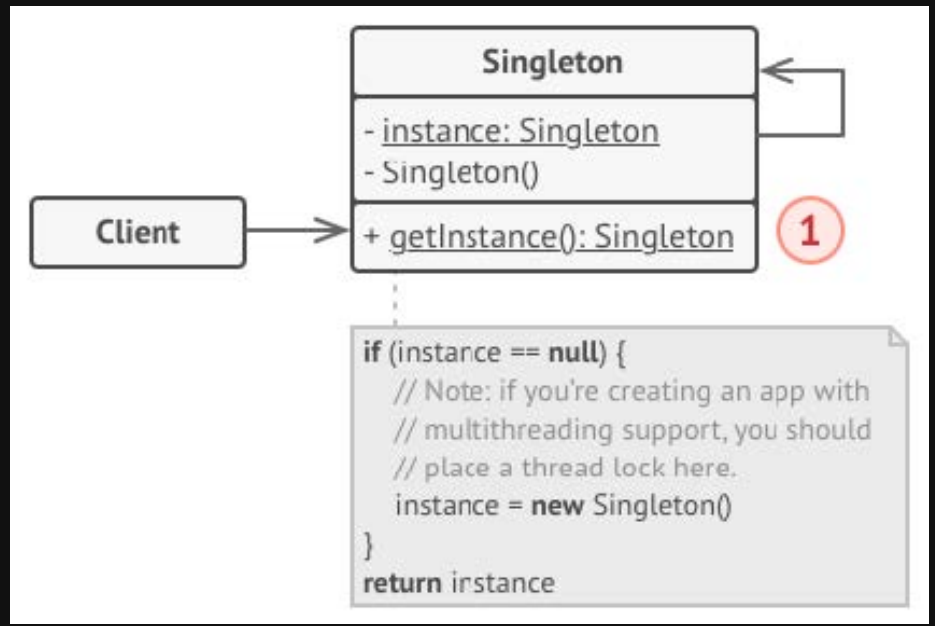
Singleton Pattern

What type of pattern?

Creational pattern

The singleton pattern ensures that a class has only one instance. It provides a global access point to this instance.

It helps avoid initialisation overhead when only 1 copy of an instance is needed.

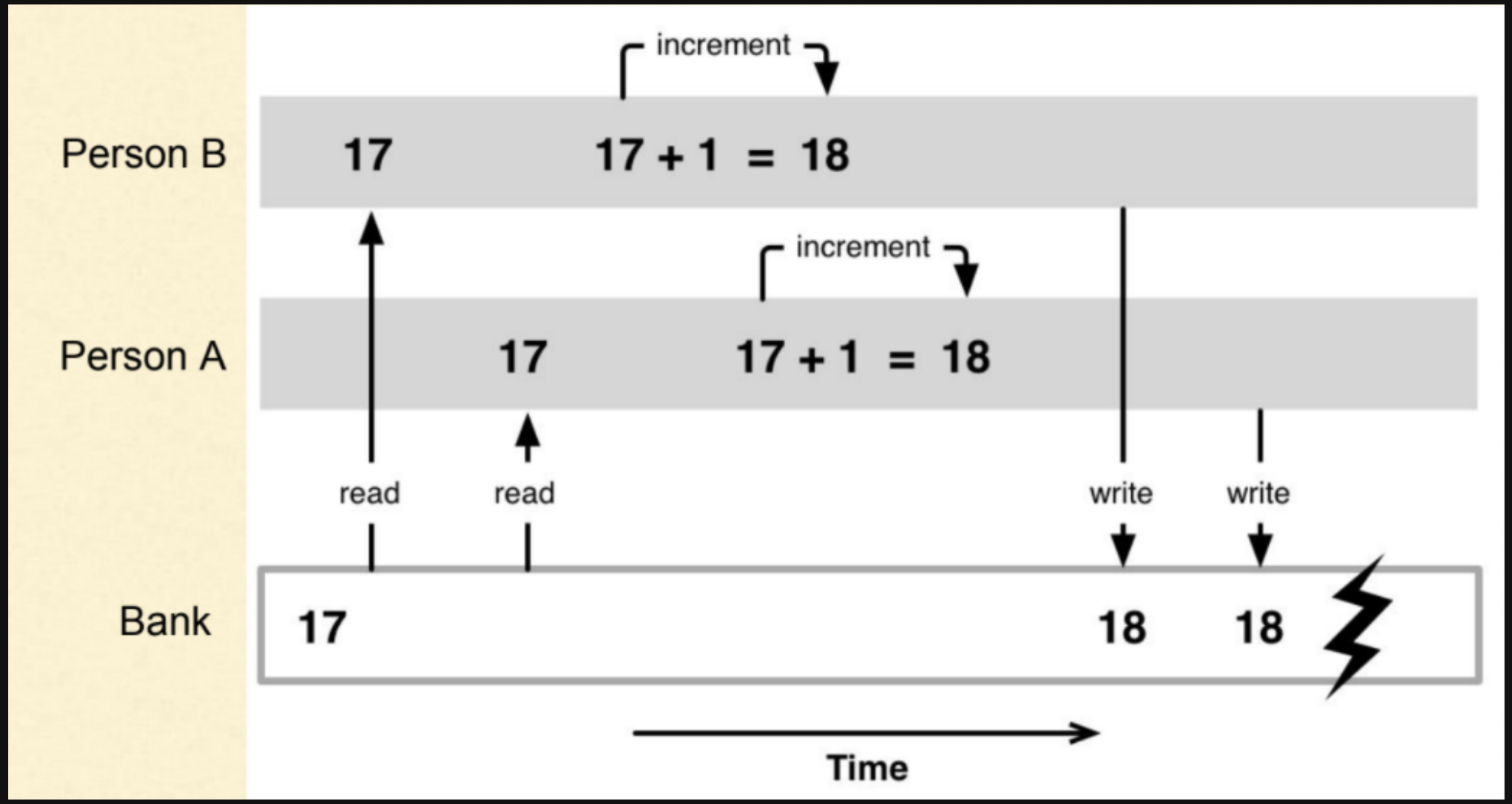


Code Demo

Heist.java

Code Demo

Link


increment
Person B
17
17 + 1 = 18
increment
Person A
17
17 + 1 = 18
read
read
write
write
Bank
17
18
18
Time

Attendance

Feedback

https://forms.gle/R4sMTTQzPC4vqXSN8