Design Patterns Final Exam – 2021/2022, Fall Semester

24-January-2022, [2 Hrs.], [40 Marks]

Answers provided by:

Fares Abuali[201810408] and Waleed Zriqui[201810582]

Question#1 [6 Marks]: Short-Answer Questions:

1. [3 Marks]

Singleton is a creational design pattern that ensures you only have one instance while providing a single point of access to this instance.

A. [2 Marks] Mention 2 drawbacks of Singleton pattern.

My Answer (Not sure about it):

1. The need to make it thread-safe causes to slow down the execution speed, because we need to lock the critical area (inside the getInstance() method).

2. The singleton might need to do more than one single responsibility, because the singleton is responsible for the creation of the object, in addition to the management of other methods. Hence, it has more than one reason to change),

Correct Answer (Modified by Mohammad Abohasan):-

1- no subclassing. ما بنقدر نعمل انهرتنس منه

2- كونه ما عنا غير انستنس واحدة وممكن يتم استعمالها من اكثر من مكان فاذا حصل فيها عطل بتخرب عند الجميع

B. [1 Mark] Singleton violates one of the principles, what is the violated principle? Explain.

It violates the Single Responsibility principle as I mentioned above.

2. [3 Marks]

A. [1 Mark] What is the principle that is violated by the following code:

```
public class Client {
    public int getPageNumber(Document document) {
        return document.getCurrentPage().getPageNumber();
    }
}
```

Answer:

The violated principle is the "Least Knowledge Principle", or (Law of Demeter: Talk only to your immediate friends)

B. [2 Marks] Refactor (Rewrite) the code so that it does not violate that principle:

Answer:

Since the Document class has attribute of type 'Page', then define a function in the Document class which will be responsible only for returning the currentPage's number. And then the Client class will easily be able to call this method using object 'document' of class 'Document', and now the Client doesn't need to know much about how to get the pageNumber (hide the details (apply least knowledge")

```
// This was my code (Fares Abuali), not 100% sure
public class Document {
    private Page currentPage;
    // Document class already has attribute called page or currentPage let's say

    public int getPageNumber(Document document) {
        return currentPage.number();
    }
}

public class Client {
    public int getPageNumber(Document document) {
        return document.getPageNumber();
    }
}
```

Question#2 [10 Marks]: Template Method:

Assume you want to design an app for restaurant that enables the customer to order meal by following certain steps: get Menu, select Meal, get Meal, pay, give a tip.
getMenu() => By default, the customer will be given a printed menu to browse, but he can ask to browse the menu using the app version instead of the printed menu version.

selectMeal() => By default, the waiter will come to the customer's table and take the customer's order, but the customer can go to the register (the cashier) and give them his order instead of asking the waiter to come.

getMeal() => The customer must wait for his meal to be delivered by the waiter. There are no other options available.

pay() => The payment method is left the customer's preference.

Tip() => By default, the customer is supposed to give a tip, but if he doesn't want to give a tip, he can decide.

1. [6 Marks]: Write the code for the Template method and its class.

Answer: My code

```
/*
* author: Fares Abuali
*/
public abstract class RestaurantTemplate {
```

```
    public final void orderMeal() {
        getMenu(); // concrete not final
        selectMeal(); // concrete not final
        getMeal(); // concrete and final
        pay(); // abstract
        if (wantsToAddTip()) {
            // hook
            tip();
        }
    }

    public void getMenu() {
        System.out.println("You will be given a printed menu to browse");
    }
    public void selectMeal() {
        System.out.println("Raise your hand and the waiter will come to your table to
take your order");
    }
    public final void getMeal() {
        System.out.println("Kindly wait until the waiter gives you your order");
    }
    public abstract void pay();
    public void tip() {
        System.out.println("Thanks for the tip");
    }
    public boolean wantsToAddTip() { return true; }

}// end class
```

2. [4 Marks]: Write the code of a concrete restaurant class that makes use of the above template class, the protocols in this restaurant will be:

- the customer will browse the menu using the app, will pay using Visa Card, and doesn't want to give a tip.

My Code Answer:

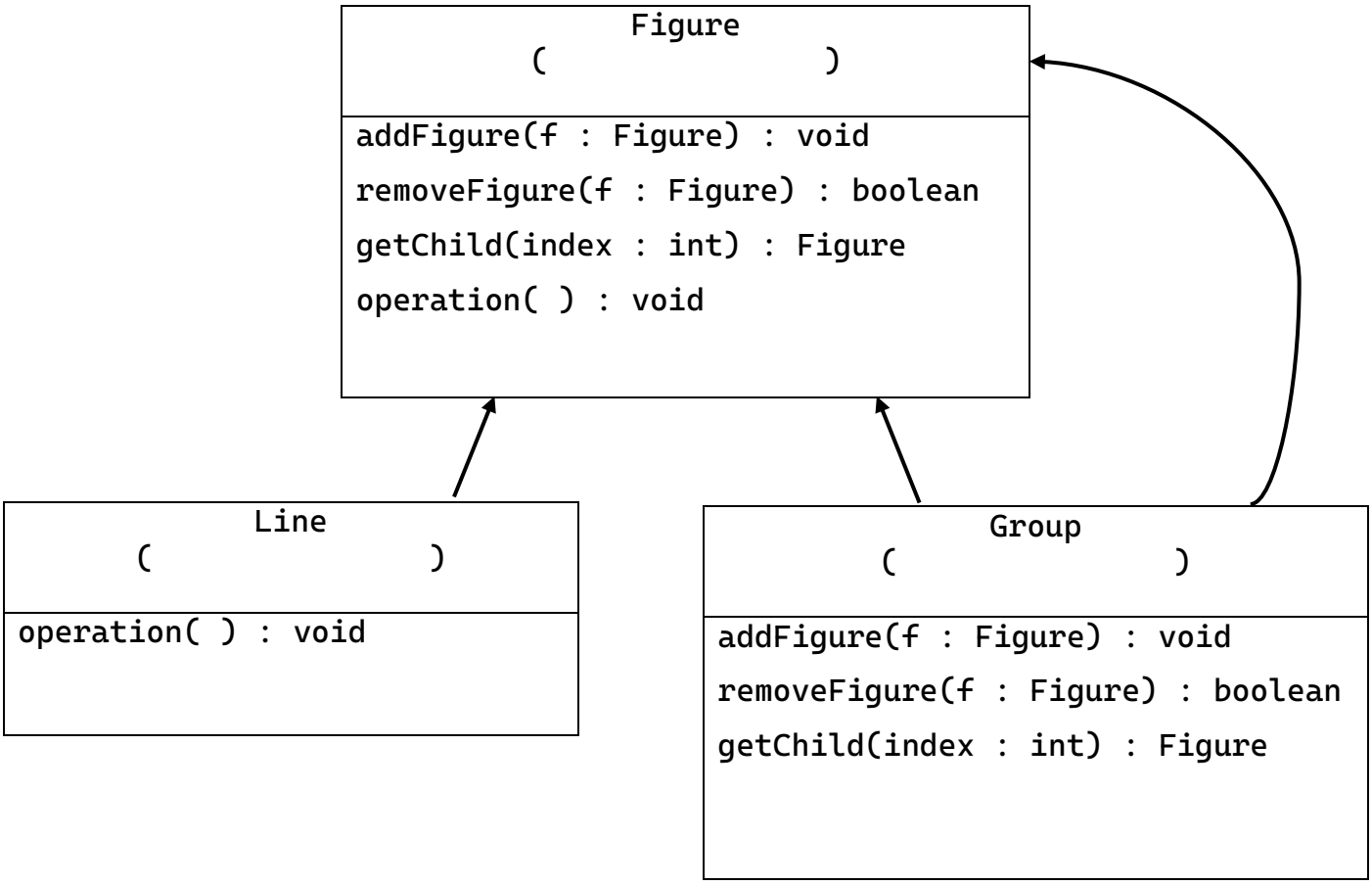
```
/*
* author: Fares Abuali
*/
public class FaresRestaurant extends RestaurantTemplate {

    @Override
    public void getMenu() {
        System.out.println("You will be browsing the menu from the app");
    }
    @Override
    public void pay() {
        System.out.println("Pay using Visa Card");
    }
```

```
    @Override
    public boolean wantsToAddTip() {
        return false;
    }
}
```

Question#3 [6 Marks]: Most Appropriate Pattern

Assume this is app that allows users to draw Figures such as lines, circles, and squares. Also user can draw multiple figures and make them as one group. Given the following class diagram design of the app:



1. [1 Mark] What is the most appropriate pattern to implement the above design?

My Answer: Composite Pattern (Composite Iterator)

2. [3 Marks] In the above parenthesis ( ), fill in the role of each class.

My Answer:
Figure is (Component), it is high level class (Abstract Component)
Note: The 'Figure' class must be abstract in order to be considered a high-level and to prevent

the client from instantiating it.
يعني كلاس الفيغر ما فيه ولا ميثود أبستراكت لكن مع ذلك لازم يضل الكلاس أبستراكت.

Line is (Leaf), which is a Concrete Component class that doesn't have children

and Group is (Composite), which is a Concrete Component class that have children of type Figure.

3. [2 Marks] Write the code of getChild(int index) method in the Figure class.

My Answer

```
public Figure getChild(int index) {
    throw new UnsupportedOperationException();
}
```

// we don't want the make the getChild in the Figure class abstract. So, make it throw an exception if any of the concrete classes tried to use it without overriding it.

Question#4 [8 Marks]: Recognize the Patterns

In the following code, there are several patterns used, please name the patterns and for each pattern, mention the name of class and its role in that pattern

```
public
interface A {

      public B
a1();

     }
```

```
public interface B {
   public void
b1(Foo f);

    public void b2();

}
```

```
public  class Bar
implements Foo {

 public void    foo1(){


System.out.println("");

      }     }
```

```
public  interface
Foo {
  public void
foo1();
      }
```

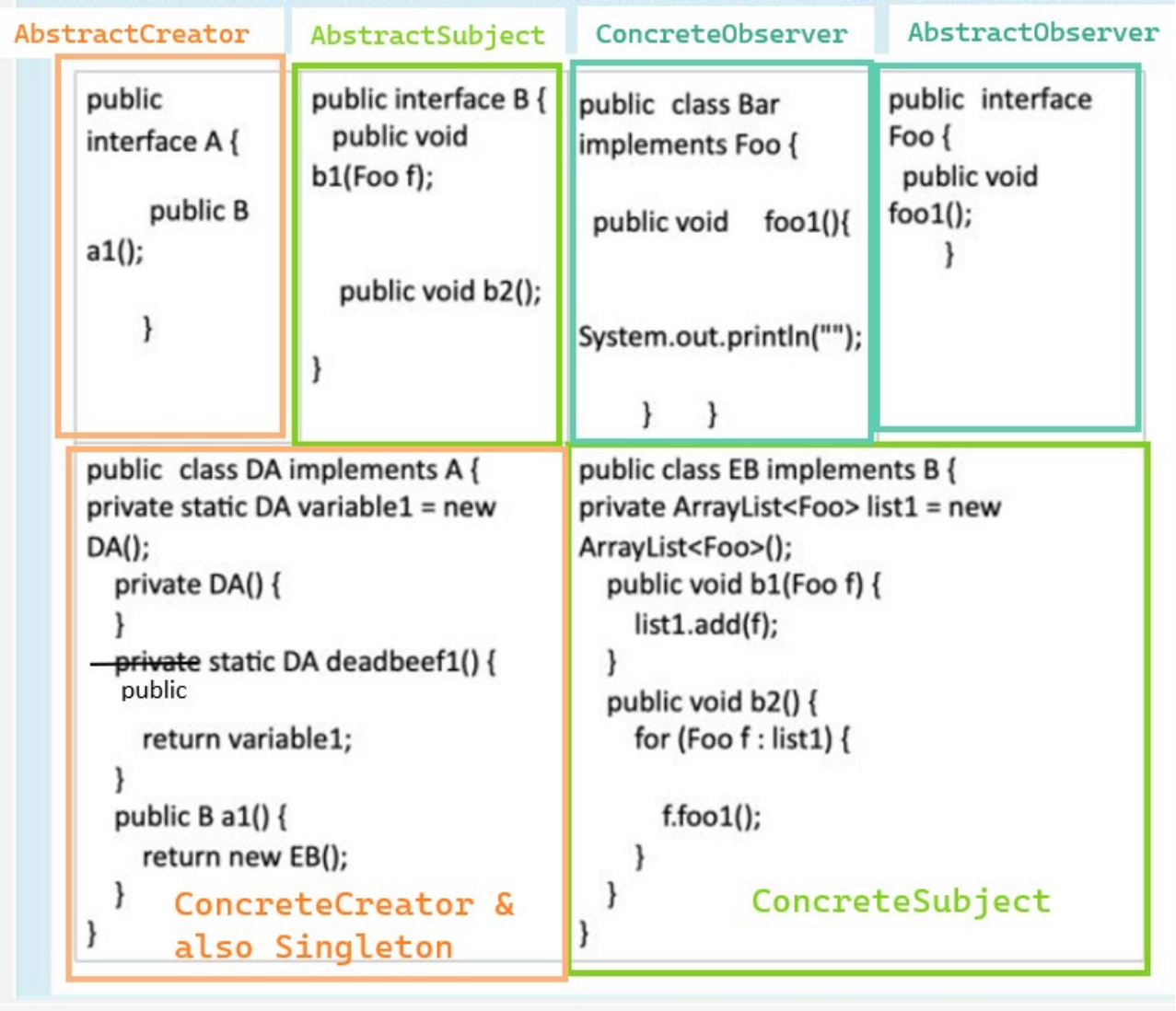
```
public  class DA implements A {
private static DA variable1 = new
DA();
   private DA() {
   }
   private static DA deadbeef1() {

      return variable1;
   }
   public B a1() {
      return new EB();
   }
}
```

```
public class EB implements B {
private ArrayList<Foo> list1 = new
ArrayList<Foo>();
   public void b1(Foo f) {
      list1.add(f);
   }
   public void b2() {
      for (Foo f : list1) {

         f.foo1();
      }
   }
}
```

My Answer: (Not 100% sure about the AbstractCreator and the ConcreteCreator)



Interface B === AbstractSubject

b1(Foo f) === registerObserver

B2() === notifyObservers()

Class EB implements B === ConcreteSubject

Interface Foo === AbstractObserver

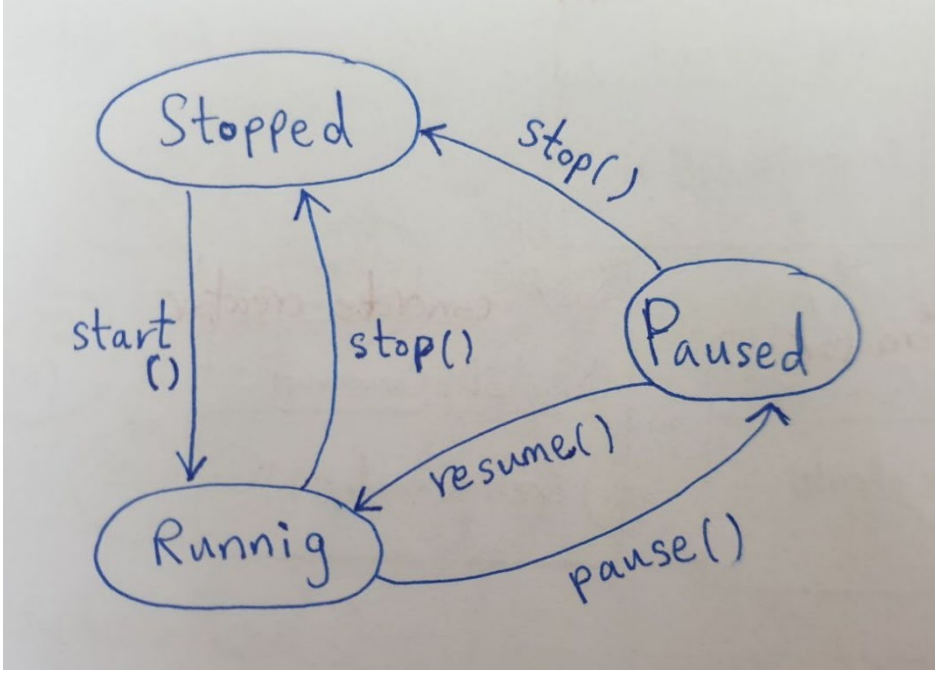
Class Bar implements Foo === ConcreteObserver

Interface A === AbstractCreator/Store/Factory

Class DA implements A === ConcreteCreator/Store/Factory & also it is a Singleton Class

Question#5 [10 Marks]: State Pattern

Given the following state diagram for a microwave machine with the states and transitions:



1. [2 Marks] Write the code for the State Interface.

My Answer: We have 3 States and 4 Transitions, so we need to declare 4 functions in each state class:

```
public interface State {
    public void start();
    public void stop();
    public void pause();
    public void resume();
}
```

2. [4 Marks] Write the code for the Microwave Machine class, show the constructor, the getters, and the setState method.

My Answer:

```
/*
* author: Fares Abuali
*/

public class MicrowaveMachine {
    private State stoppedState;
    private State runningState;
    private State pausedState;

    private State state; //current state
```

```
    // The constructor
    public MicrowaveMachine() {
        this.stoppedState = new StoppedState(this);
        this.runningState = new RunningState(this);
        this.pausedState = new PausedState(this);

        this.state = stoppedState; //initial state
    }
    // The getters
    public State getStoppedState() {

        return this.stoppedState;
    }
    public State getRunningState() {
        return this.runningState;
    }
    public State getPausedState() {
        return this.pausedState;
    }
    // The setter
    public void setState(State state) {
        this.state = state;
    }
    // delegate to the state classes:
    public void start() {
        state.start();
    }
    public void stop() {
        state.stop();
    }
    public void pause() {
        state.pause();
    }
    public void resume() {
        state.resume();
    }
}
```

3. [4 Marks] Write the code for the Stopped State class.

My Answer:

```
public class StoppedState implements State {
    MicrowaveMachine microwave;

    public StoppedState(MicrowaveMachine micro) {
        this.microwave = micro;
    }

    @Override
    public void start() {
```

```
        System.out.println("The microwave is about to start..");
        microwave.setState(microwave.getRunningState());
        // Stopped ---> Running
    }

    @Override
    public void stop() {
        System.out.println("The microwave already OFF!");
    }

    @Override
    public void pause() {
        System.out.println("The microwave already OFF!");
    }

    @Override
    public void resume() {
        System.out.println("You should start the microwave first..");
    }
}
```

Best Wishes :)

Fares H. Abuali 24-Jan-2022